INCL. 25 PAGINA'S AMIGA

Commodore INFO

Prijs ƒ 7,95 / Bfr. 160

8 710966 001332

ONAFHANKELIJK BLAD VOOR COMMODORE GEBRUIKERS

Jaargang 6, NO.7, nov./dec. 1989

LISTINGS:

- Briefmaker 128
- Superchange 128
- Collis 16
- Apenspel 64
- Spreadsheet 64
- Galgje 64
- Lettertekenaar 64

NIEUWE SERIE: 68000 INTERN

KLEUREN AANPASSEN OP UW SCHERM

KOALA PAINTER

GRAPHICS OP DE C-64

DISKDRIVES C-128

TURBOBOARDS VOOR DE AMIGA

2286 AT-KAART

ANIMAGIC

Inclusief Gratis kaartje Computer Info beurs 4 nov. RAI Amsterdam

Vaste rubrieken:

- Tips en Trucs voor de C-64
- Amiga Tips
- Geos Info
- Amiga DOS
- Basic Basic
- Kleine advertenties

Commodore Info

Verschijnt 8x per jaar
Jaarg.6, no.7, nov/dec. 1989

Uitgave:
Sala Communications
Uitgever:
Vic Sharfman
Redactie:
Ir. L. Sala hoofdredacteur
J. Bodzinga adj. hoofdred.
drs. J. Boers eindredacteur
H. Smeenk, drs. U. Schuurmans, R. Goudriaan, B. Munniksma, B. Venema, P. Boncz, MGCC/Johan & Johan
drs. H. Zoete produktie/planning
M. van Kaathoven produktie
J. Broekhuizen produktie

Redactiesecretariaat:
R. van Zalingen
Strip: Bert Tier
Illustraties: Ben van Mierlo

Advertentie-exploitatie:
Ing. V. Sala, Ing. B. Sala,
D. van Vlijmen
Weesperstraat 103
1018 VN Amsterdam
tel. 020-273198
Redactie adres:
Postbus 43048
1009 ZA Amsterdam
tel. 020-228871
Listingtelefoon: (ma: 17.00-21.00)
02155-25162

Abonnementen en administratie:
Sandra v.d. Meyden en Marjo Jansen
Postbus 43048
1009 ZA Amsterdam
tel. 020-248006
Vragen betreffende abonnementen ontvangen wij bij voorkeur schriftelijk, met meesturen van het omslagetiket.
Abonnement:
Voor 8 nummers *f* 47,50 of Bfr. 975 per jaar. Betaling op giro 1585491 (België: BBL nr. 310050602562) t.n.v. SAC/Commodore-Info. Oude nummers kunt U alleen krijgen bij vooruitbetaling van *f* 6,75 op de bovenstaande rekening. Ook telefonische opgave voor een abonnement is mogelijk. Bel GRATIS 06-02242222 (teleservice), elke dag tot 20.20 uur (dus ook in het weekend). België: 115555, dagelijks tot 22.00 uur. Deze telefoonnummers zijn alleen bedoeld voor opgave van NIEUWE abonnementen.
Opzegging dient schriftelijk te geschieden uiterlijk twee maanden voor de aanvang van een nieuwe abonnementsperiode van een jaar.

Omslagfoto: Met dank aan Altycos

Zetwerk & druk: NDB, Zoeterwoude
Distributie:
In Nederland: Betapress, Gilze
In België: AMP, Brussel

ISSN: 0169-3085

Inhoud van dit nummer

Redactioneel

Zoals u zoaanstonds zult aanschouwen is Commodore Info dit maal in een nieuw jasje gestoken. In deze tijd waarin de Commodore markt volop in beweging is, moet een blad ook met zijn tijd mee. De vormgeving is wat strakker geworden en wij hopen hiermee ook de leesbaarheid van de artikelen te vergroten, zodat u met meer plezier de volgende Commodore Info's zult lezen. Naast de veranderde vormgeving zal het u ongetwijfeld opvallen dat het aantal listings, in het hartkatern met enige pagina's teruggebracht is. De reden hiervoor is dat we door deze inkrimping meer ruimte overhouden om andere artikelen te kunnen plaatsen. De laatste jaren is er een steeds grotere vraag gekomen naar informatie over de Amiga machines. Door het scheppen van wat extra ruimte in het blad, kunnen wij aan deze vraag voldoen. Meer Amiga dus en minder listings. Wij hopen dat het u zal bevallen.

Verander zelf de kleuren op uw scherm

Color Adjuster een handige routine voor kleuraanpassing

Programmatuur ontwikkelen is leuk en wordt veel leuker als je een doel voor ogen hebt. Daarom hier een routine, geschreven in C, om de kleuren van het WorkBench scherm te veranderen. Opstartbaar vanuit de startup-sequence en compatible met AmigaDOS (tjongejonge wat een geleuter!)

Is het u al eens overkomen? U wilde de kleuren van het WorkBench scherm of willekeurig ander scherm veranderen maar u beschikte niet over een programma om dit te doen. Daarom hier een kleine routine om dit voor u te verwezenlijken. Het programma detecteert zelf in welk scherm u aan het werk bent en over hoeveel kleuren dit scherm beschikt. U kunt de routine dus in elk willekeurig scherm activeren, ervan uitgegaan dat u dit scherm geactiveerd heeft, door hier met de muis in te klikken. Toch kan het gebeuren dat u problemen met deze routine krijgt, maar daarover straks meer.

Hoe werkt het?

De grote vraag luidt; hoe met het programma om te gaan? Logischerwijs dient het eerst opgestart te worden. Wilt u gewoon de kleuren van het WorkBench scherm veranderen dat typt u vanuit CLI in *coladjust.* Het programma zal een requester tevoorschijn toveren met een aantal gadgets. Wilt u in een ander scherm de kleuren veranderen dan zult u enige tijd nodig hebben om het andere scherm met de muis weer te activeren. Het is dus noodzaak een vertraging op te wekken alvorens het programma actief wordt. Typ hiervoor het volgende batchbestand met een Editor, bijvoorbeeld *Ed,* onder de naam *ColOpstart* in.

```
wait 5 secs
coladjust
```

Save het bestand en verlaat de editor. Voer het batchbestand vervolgens uit met behulp van het *Execute* commando.

Execute *ColOpstart*

Het batchbestand zorgt er dan voor dat het 5 seconden duurt voordat het programma opstart. In de tussentijd kunt u het gewenste scherm activeren.

Maar hoe ziet het er uit?

Allereerst ziet u drie staande staafgadgets met er boven vermeldt, 'R', 'G' en 'B'. Zoals u al vermoedt, ze zijn bedoeld om de afzonderlijke kleurcomponenten Rood, Groen en Blauw te veranderen. Verschuif deze gadgets en het kleurcomponent veranderd mee. Rechts van de staafgadgets vindt u, afhankelijk van het aantal kleuren in het scherm, een aantal gadgets waarmee de kleur te selecteren is. Wilt u bijvoorbeeld de kleuren van het WorkBench scherm veranderen, dan zullen hier 4 gadgets aanwezig zijn. Kies één van deze gadgets door er op te klikken, en het vlak beneden de staafgadgets zal met de gekozen kleur mee veranderen. Tot slot drie gadgets die tegen de onderrand geplaatst zijn, genaamd 'Reset', 'Nieuw' en 'OK'. Het *Reset* Gadget dient om de kleur, welke u op het moment aan het veranderen bent, weer terug te zetten naar zijn originele kleur. Het geldt dus alleen voor de kleur waarmee u bezig bent, niet voor de andere kleuren. De andere kleuren kunt u weer herstellen met *Nieuw.* Alle kleuren worden daardoor weer naar hun oorspronkelijke kleurwaarden teruggezet. Tot slot nog *OK.* Het mag duidelijk zijn, hiermee geeft u te kennen dat het



nieuwe kleurpalet OK is en gebruikt mag worden. Het requester verdwijnt hierdoor, u komt weer in DOS. Nog een opmerking. Als u een kleur wilt aanpassen, maar u heeft nog geen kleur uitgekozen, dient u eerst een kleur te kiezen anders zal het programma niets doen!!

Het programma in detail

Het programma bestaat uit vier functies te weten, *Init(), main(), RememberClrs() en AdjustProps().*
De functie Init() zorgt ervoor dat alle variabelen juist geinitialiseerd worden. Het roept de functie RemeberClrs() aan om te zorgen dat de originele kleuren worden opgeslagen, het reserveert geheugen voor bepaalde schermdata en het legt de laatste links tussen de Gadget en Image structuren. In main(), het eigenlijke hoofdprogramma, worden allereerst enkele fitale libraries geopend. Vervolgens wordt aan RImage een adres toegekend waar voldoende geheugen aanwezig is voor drie keer de groote van de Image structuur. Hier worden de grafische data voor de drie R,G en B staven opgeslagen. Vervolgens wordt met de functie ModifyIDCMP() het huidige venster aangepast



Steeds meer wordt gebruik gemaakt van de mogelijkheden van het kleurenscherm

voor gadgetinvoer. Als dit voorbij is wordt de functie Init() aangeroepen voor de laatste initialisaties. Tenslotte wordt dan, als alles goed gegaan is, het requester geopend met de functie Request.
We komen nu in de 'mainloop' van het programma terecht. Dit is het deel waar de verschillende gadgets gecontroleerd worden op activering. Zolang er geen gadget ingedrukt wordt zal het programma blijven hangen in de Wait() functie. Voor de precieze werking van deze functie kunt u beter een Amigahandboek erop naslaan, het komt er in ieder geval op neer dat zodra er een gadget ingedrukt wordt, het programma weer begint te lopen. Daarna nog een keer een controle of er werkelijk data via een MessagePort binnengekomen is. Is dit het geval dan wordt het Messagetype opgeslagen in Class en de verwijzing naar de Gadget structuur wordt aan GadgetPtr toegekend. De message wordt weer terugverzonden en het programma begint de kijken welk gadget om aandacht vroeg. Er zijn twee soorten gadgets in het programma aanwezig.

- 1. De drie balken die vanaf het moment dat ze ingedrukt worden vereisen constante aandacht
- 2. De overige gadgets die pas aandacht nodig hebben als ze weer losgelaten worden.

De gadgets genoemd onder punt 1 zijn van het type GADGETDOWN. Zo gauw ze worden ingedrukt krijgt het programma een mededeling. Gadgets genoemd onder punt 2 behoeven pas aandacht als de muisknop weer losgelaten wordt. Vandaar de benaming GADGETUP.
Gadget 4 en 5, de 'Reset' en 'Nieuw' gadgets zijn van het type GADGETUP evenals de kleuren gadgets die waarden groter dan 6 hebben. Gadgets 1, 2 en 3 zijn de staven en zijn dus van het type GADGETDOWN. Het programma besteedt aandacht aan de gadgets zolang de flags van de gadgets aangeven dat ze SELECTED zijn. Tot slot gadget nummer 6. Deze wordt gecontroleerd in de allesomvattende *do-while()* lus. Zolang dit gadget niet ingedrukt wordt blijft het programma in deze lus. Het requester wordt door *Intuition* zelf weer van het scherm verwijdert doordat in de declaraties, in het eerste deel van de listing, onder OKGadget aangegeven wordt dat dit gadget van het type ENDGADGET is.

Tot slot de functie AdjustProps(). Deze functie verandert de positie van het schuifelement in de staven, afhankelijk van de kleuren welke deze staven vertegenwoordigen.

Eigen inbreng

Zoals u ziet is een dergelijk programma niet zo moeilijk te programmeren. Verander zelf wat aan het programma. Begin er mee te experimenteren. Maak bijvoorbeeld een routine waarmee de kleurdata vanuit een batchbestand uit DOS in te lezen zijn of maak een spreadfunctie in het programma zoals bijvoorbeeld ook D'Paint kent. Stuur deze veranderingen naar ons op zodat we iedereen hiervan kunnen laten genieten.
Wilt u meer informatie over gadgets, over een paar maanden hebben we een aflevering van de C cursus die gewijd zal zijn aan gadgets. Wilt u hier niet op wachten, ook in de boekhandel zijn goede boeken aanwezig die genoeg vertellen over gadgets. Een voorbeeld, Het grote Amiga C-boek van Data Becker,
ISBN 90 229 3483 7.

Johan & johan.

Listing van ColAdjuster

Instructies:

Programmanaam	: ColAdjuster.c
Programmeertaal	: C
Gebruikt programma	: Aztek C v3.6
Opties	: cc ColAdjuster.c +L
	: ln ColAdjuster.o -lc32
Aanroep	: ColAdjuster

```
#include "functions.h"
#include "exec/types.h"
#include "exec/memory.h"
#include "intuition/intuitionbase.h"
#include "graphics/view.h"
#include "graphics/gfxbase.h"
#include "graphics/gfxmacros.h"

#define RP CR.RWindow-> WScreen->RastPort
#define VPORT ModWindow->WScreen->ViewPort
#define CMAP ModWindow->WScreen->ViewPort.ColorMap
SHORT SelPairs[] = {0,0,55,0,55,20,0,20,0,0 };
SHORT CPairs2[] = {0,0,319,0,319,149,0,149, 0,0 };
SHORT CPairs1[] = {0,0,311,0,311,145,0,145, 0,0 };
SHORT ColPairs[] = { 0,0,80,0,80,15,0,15,0,0 };
UWORD OrgColor[64];

struct IntuitionBase *IntuitionBase;
struct GfxBase *GfxBase;
struct Window *ModWindow;
struct ViewPort *ModViewPort;
struct ColorMap *ModColorMap;
struct Image *RImage;
struct Image ClrImage[64];
struct Image CurImage;
struct PropInfo BInfo = {
      AUTOKNOB|FREEVERT,
      0,4369,0,4369,
      10,10,NULL,NULL,
      NULL,NULL };
struct PropInfo GInfo = {
      AUTOKNOB|FREEVERT,
      0,4369,0,4369,
      10,10,NULL,NULL,
      NULL,NULL };
struct PropInfo RInfo = {
      AUTOKNOB|FREEVERT,
      0,4369,0,4369,
      10,10,NULL,NULL,
      NULL,NULL };
struct Border OKBorder = {
      0,0,0x01,0x01,JAM1,5,SelPairs,NULL };
struct Border CanBorder = {
      0,0,0x01,0x01,JAM1,5,SelPairs,NULL };
struct Border ResBorder = {
      0,0,1,0,JAM1,5,SelPairs,NULL };
struct Border Color2Border = {
      0,0,0x01,0x01,JAM1,5,
      CPairs2,NULL };
struct Border Color1Border = {
      4,2,0x01,0x01,JAM1,5,
      CPairs1,&Color2Border };
struct Border ColBorder = {
      0,0,0x01,0x01,JAM1,5,ColPairs,NULL };

struct IntuiText OKText = {
      0x01,0x00,JAM1,22,7,
      NULL,(UBYTE *)"Ok",NULL };
struct IntuiText CanText = {
      0x01,0x00,JAM1,5,7,
      NULL,(UBYTE *)"Nieuw",NULL };
struct IntuiText ResText = {
      0x01,0x00,JAM1,7,7,
      NULL,(UBYTE *)"Reset",NULL };
struct IntuiText RText = {
      0x01,0x00,JAM1,4,-8,
      NULL,(UBYTE *)"R",NULL };
struct IntuiText GText = {
      0x01,0x00,JAM1,4,-8,
      NULL,(UBYTE *)"G",NULL };
struct IntuiText BText = {
      0x01,0x00,JAM1,4,-8,
      NULL,(UBYTE *)"B",NULL };
struct Gadget ClrGadget[64];
struct Gadget CUGadget = {
      NULL,0,0,25,7,
      GADGHBOX|GADGIMAGE,RELVERIFY,
      BOOLGADGET|REQGADGET,NULL,
      NULL,NULL,NULL,NULL,NULL,NULL };
struct Gadget OKGadget = {
      NULL,140,120,55,20,GADGHCOMP,
      RELVERIFY|ENDGADGET,BOOLGADGET|REQGADGET,
      (APTR)&OKBorder,
      NULL,&OKText,
      NULL,NULL,6,NULL };
struct Gadget CanGadget = {
      &OKGadget,75,120,55,20,GADGHCOMP,
      RELVERIFY,BOOLGADGET|REQGADGET,
      (APTR)&CanBorder,
      NULL,&CanText,
      NULL,NULL,5,NULL };
struct Gadget ResGadget = {
      &CanGadget,10,120,55,20,GADGHCOMP,
      RELVERIFY,BOOLGADGET|REQGADGET,
      (APTR)&ResBorder,
      NULL,&ResText,
      NULL,NULL,4,NULL };
struct Gadget RGadget = {
      &ClrGadget[0],130,15,20,80,GADGHCOMP,
      GADGIMMEDIATE,PROPGADGET|REQGADGET,
      NULL,NULL,&RText,NULL,
      (APTR)&RInfo,3,NULL };
struct Gadget GGadget = {
      &RGadget,70,15,20,80,GADGHCOMP,
      GADGIMMEDIATE,PROPGADGET|REQGADGET,
      NULL,NULL,&GText,NULL,
      (APTR)&GInfo,2,NULL };
struct Gadget BGadget = {
      &GGadget,10,15,20,80,GADGHCOMP,
      GADGIMMEDIATE,PROPGADGET|REQGADGET,
      NULL,NULL,&BText,NULL,
      (APTR)&BInfo,1,NULL };
struct Requester CR = {
      NULL,40,10,320,150,0,0,&BGadget,
      &Color1Border,NULL,NULL,2,
      NULL,NULL,NULL };
UWORD *ClrData;
UWORD ImageData[] = {
      0xffff,0xffff,0xffff,0xffff,
      0xffff,0xffff,0xffff,0xffff,
      0xffff,0xffff,0xffff,0xffff,
      0xffff,0xffff,0xffff,0xffff,
      0xffff,0xffff,0xffff,0xffff };

Init()
{
 register int i,l;
 USHORT Depth,Colors,Width;

 Depth=ModWindow->WScreen->BitMap.Depth;
 Colors=1<<Depth;
 RememberClrs(Colors);
 ClrData=(UWORD *)

     AllocMem(sizeof(UWORD)*32*(Depth+1),MEMF_CHIP|MEMF_CLEAR
     );

 for(i=0;i<Depth;i++)
   {
    for(l=0;l<20;l++)
      *(ClrData+l+(16*i)) = ImageData[l];
   }
 Width=230+((Colors/15)*35);
 CPairs2[2]=CPairs2[4]=Width-1;
 CPairs1[2]=CPairs1[4]=Width-9;

 for(i=0;i<Colors;i++)
   {
```

```
     ClrImage[i].LeftEdge=0;
     ClrImage[i].TopEdge=0;
     ClrImage[i].Width=25;
     ClrImage[i].Height=7;
     ClrImage[i].Depth=Depth;
     ClrImage[i].ImageData=ClrData;
     ClrImage[i].PlanePick=(UBYTE)i;
     ClrImage[i].PlaneOnOff=(UBYTE)i;
     ClrImage[i].NextImage=NULL;

     ClrGadget[i]=CUGadget;
     ClrGadget[i].LeftEdge=190+(int)(i/15)*35;
     ClrGadget[i].TopEdge=5+((i&15)*8);
     ClrGadget[i].GadgetRender=(APTR)&ClrImage[i];
     ClrGadget[i].GadgetID=7+i;
     ClrGadget[i].NextGadget=&ClrGadget[i+1];
    }
 ClrGadget[i].NextGadget=&ResGadget;
 CR.Width=Width;
 return(Depth);
}

main()
{
 register int l;
 UBYTE succes;
 UWORD color;
 ULONG Flags,Class;
 USHORT
    GadgetID,CurColor,BComp,GComp,RComp,RMod,GMod,BMod,Depth
    ;
 struct IntuiMessage *ModMes;
 struct Gadget *GadgetPtr;
 if(!(IntuitionBase=(struct IntuitionBase *)
    OpenLibrary("intuition.library",0L)))
      {
       printf("Geen intuition.library!\n");
       exit(FALSE);
      }
 if(!(GfxBase=(struct GfxBase *)
    OpenLibrary("graphics.library",0L)))
      {
       printf("Geen graphics.library!\n");
       CloseLibrary(IntuitionBase);
       exit(FALSE);
      }
 if(!(RImage=(struct Image *)
   AllocMem(3*sizeof(struct Image),MEMF_CHIP|MEMF_CLEAR)))
      {
       printf("Geen Image geheugen meer!\n");
       CloseLibrary(GfxBase);
       CloseLibrary(IntuitionBase);
       exit(FALSE);
      }
 RGadget.GadgetRender=(APTR)RImage;
 GGadget.GadgetRender=(ATPR)RImage+sizeof(struct Image);
 BGadget.GadgetRender=(APTR)RImage+2*sizeof(struct Image);
 Flags=IntuitionBase->ActiveWindow->IDCMPFlags;
 ModWindow=IntuitionBase->ActiveWindow;
 ModifyIDCMP(ModWindow,Flags|GADGETUP|GADGETDOWN);

 Depth=Init();
 succes=Request(&CR,ModWindow);

 do
   {
    Wait(1<<ModWindow->UserPort->mp_SigBit);
    while((ModMes=(struct IntuiMessage *)
       GetMsg(ModWindow->UserPort))!=NULL)
        {
         Class=ModMes->Class;
         GadgetPtr=(struct Gadget *)ModMes->IAddress;
         ReplyMsg(ModMes);
        }
    GadgetID=GadgetPtr->GadgetID;
    if(Class==GADGETUP)
       {
        switch(GadgetID)
          {
           case 4:
              BComp=color & 0x0f;
              GComp=(color>>4) & 0x0f;
              RComp=(color>>8) & 0x0f;
              SetRGB4(&VPORT,CurColor,RComp,GComp,BComp);
              AdjustProps(RComp,GComp,BComp);
              break;

          case 5:
              LoadRGB4(&VPORT,OrgColor,1<<Depth);
              RemakeDisplay();
              color=GetRGB4(CMAP,CurColor);
              BComp=color & 0x0f;
              GComp=(color>>4) & 0x0f;
              RComp=(color>>8) & 0x0f;
              AdjustProps(RComp,GComp,BComp);
              break;
          }
    if(GadgetID>6)
       {
        SetAPen(&RP,GadgetID-7);
        RectFill(&RP,80,110,160,125);
        SetAPen(&RP,(GadgetID-7)^2+1);
        DrawBorder(&RP,&ColBorder,80,110);
        CurColor=GadgetID-7;
        color=GetRGB4(CMAP,CurColor);
        BComp=color & 0x0f;
        GComp=(color>>4) & 0x0f;
        RComp=(color>>8) & 0x0f;
        AdjustProps(RComp,GComp,BComp);
       }
   }
 if(Class==GADGETDOWN)
   {
    switch(GadgetID)
      {
       case 1:
        while(BGadget.Flags==SELECTED)
          {
           BMod=BInfo.VertPot/4369;
           SetRGB4(&VPORT,CurColor,RComp,GComp,BMod);
           BComp=BMod;
          }
        break;
       case 2:
        while(GGadget.Flags==SELECTED)
          {
           GMod=GInfo.VertPot/4369;
           SetRGB4(&VPORT,CurColor,RComp,GMod,BComp);
           GComp=GMod;
          }
        break;
       case 3:
        while(RGadget.Flags==SELECTED)
          {
           RMod=RInfo.VertPot/4369;
  SetRGB4(&VPORT,CurColor,RMod,GComp,BComp);
           RComp=RMod;
          }
        break;
      }
   }
  } while(GadgetID!=6)
 FreeMem(ClrData,sizeof(UWORD)*32*(Depth+1));
 FreeMem(RImage,3*sizeof(struct Image));
 CloseLibrary(GfxBase);
 CloseLibrary(IntuitionBase);
}
RememberClrs(Colors)
USHORT Colors;
{
 register int i;
 for(i=0;i<Colors;i++)
   OrgColor[i]=(UWORD)GetRGB4(CMAP,i);
}

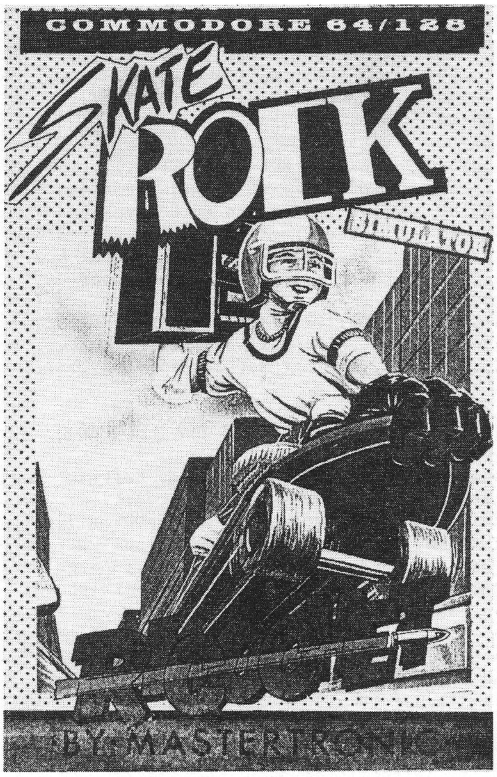
AdjustProps(RComp,GComp,BComp)
USHORT RComp,GComp,BComp;
{
ModifyProp(&BGadget,ModWindow,NULL,AUTOKNOB|FREEVERT,0,(4369*
     BComp),0,4369);

     ModifyProp(&GGadget,ModWindow,NULL,AUTOKNOB|FREEVERT,0,(
     4369*GComp),0,4369);

     ModifyProp(&RGadget,ModWindow,NULL,AUTOKNOB|FREEVERT, 0,
     (4369*RComp),0,4369);
}
```



GAMES nader bekeken...



Van Encore, software van Elite Systems Limited, ditmaal een viertal cassettes met fraaie, goedkope en toch onderhoudende games. Voor nog geen 15 gulden zijn deze spelletjes te koop en je kan er uren spelplezier aan beleven. Op het eerste gezicht doen de games eenvoudig aan, als je het vergelijkt met grote namen op softwaregebied. Maar nadat je een game hebt geladen wordt al gauw duidelijk, dat wanneer je het einde van het spel wilt bereiken, je van goede huize moet komen.

Veel vernuft voor weinig geld

SABOTEUR II.

Zoals de naam reeds doet vermoeden betreft dit spel een vervolg op SABOTEUR I. Er dienen bij dit spel 14 missies gecompleteerd te worden, waarbij jij als het mooie zusje van de Ninja, wraak neemt op haar fataal gewonde broer. In SABOTEUR I brak de Ninja in een beveiligd gebouw en stal daar een diskette met belangrijke gegevens met namen van belangrijke rebellen. Deze diskette bevatte eveneens informatie over de grote rakettensilo van de dictator. Jouw missie is, al hangend aan een "deltawing", te infiltreren in het nog steeds zwaar bewaakte gebouw, dat bestaat uit verschillende complexen, die zelfs onder de grond doorlopen. Meer dan 700 verschillende schermen zijn hierbij mogelijk. Je bent op zoek naar stukjes ponsband computertape, welke de gegevens over de vliegroutes van de raketten controleert. Zorg ervoor dat je alle stukjes bijelkaar krijgt voor het afvuren van de projectielen zal plaatsvinden. Als je eenmaal alle stukjes vergaard hebt, vlucht je op de je motor, via de enige tunnel naar buiten. Dan pas is je missie volbracht. Het lijkt allemaal eenvoudig, maar de gebouwen worden 's nachts zwaar bewaakt door grote android bewakers met vlammenwerpers. Verder kom je grote vampier-vleermuizen en zwarte panters tegen, die je weg door dit labyrint van gangen bemoeilijken. Een zeer onderhoudend spel, dat je na een paar keer pas goed onder de knie begint te krijgen. Je zult zeker in het begin een aantal keren opnieuw moet beginnen om de juiste weg te kunnen vinden. Met de diverse vechttechnieken, die je beheerst, zul je de obstakels onderweg moeten uitschakelen. Alles wat je onderweg tegenkomt, in de vorm van een kist of iets dergelijks, moet je onderzoeken. Pas dan kun je alle delen tot één geheel weten te maken. Een niet echt gemakkelijk spel. De graphics zijn niet echt wat je noemt spectaculair, maar dat is op zich niet zo erg, want dan kun je ook niet zo gauw afgeleid worden. Alle concentratie is nodig om in het doolhof van gangen en vertrekken je weg te vinden, zonder uitgeschakeld te worden.

DEEP STRIKE

Het tweede spel uit de Encore reeks, is het "vliegende spektakel" van Deep Strike, helaas een niet zo geavanceerd spel, als de verpakking doet vermoeden. De bedoeling is dat je in een éénmotorige dubbeldekker bescherming biedt aan vier bommenwerpers. Het geheel speelt zich af tijdens de Eerste Wereldoorlog. Je bekijkt het scenario vanuit de cockpit. Je ziet dan een landschap waaruit vijandelijke vliegtuigen jouw bommenwerpers belagen. Ook luchtballonnen en tanks vanaf de grond bedreigen jouw vliegtuigen. Je moet wel over een snel reactievermogen beschikken, want de vijand komt in grote getale op je af. In het wilde weg schieten is er echter ook niet bij, want af en toe gaat één van je eigen bommenwerpers voor je vliegen en als je dan aan het schieten bent is het snel bekeken. De besturing kan zowel met het toetsenbord als met de joystick plaatsvinden. Een vrolijk deuntje begeleidt het gehele spel. Als je op de M-toets drukt krijg je een een landkaart te zien met de positie van jouw squadron. Al met al niet zo'n spectaculair spel, maar voor de liefhebber van flightsimulators misschien een leuke afwisseling, of zoals het hoesje van de cassette zegt "een must voor alle liefhebbers van een goede dogfight".

STORM WARRIOR

Het volgende spel, waarbij echter wel weer mooie graphics zijn gemaakt, is een gevecht tegen de elementen om het Koninkrijk te redden van de vloek van de lelijke heks. Bij dit spel is een behendig gebruik van het zwaard noodzakelijk, omdat je anders ten prooi valt aan de vele belagers die je op je weg tegenkomt. De slechte heks heeft een vloek uitgesproken over het Koninkrijk, welke inhoudt dat het meer dan honderd jaar zal stormen. Hevige regens, huilende winden, bliksem en donder hebben al dood en verderf ge-



Veel plezier voor weinig geld

zaaid over het land. Alleen wanneer haar kwaadwillige spel verbroken kan worden zullen de verschrikkingen eindigen. Jij, als de Prins van het Koninkrijk, moet het volk redden van deze verschrikkelijke betovering. Alleen dan keert de rust en de vrede weer terug in hun levens. Gewapend met slechts een zwaard moet je het kasteel van de verschrikkelijke heks zien te vinden. Op je weg kom je eveneens zwaardvechters tegen en belagen de elementen der natuur je met hun kracht. Vliegende stenen komen uit het niets op en als ze je raken neemt je energie snel af. Je moet in dit spel behendig en slim zijn. Een niet al te gemakkelijk spel. Het valt niet echt mee om ver te komen, laat staan bij het kasteel van de heks. Onder in het beeldscherm loopt de "dood" met je mee om aan te geven hoeveel energie je nog over hebt.
het verdient aanbeveling om voor je je reis begint eerst de mogelijkheden van je zwaard uit te proberen. Kortom, een leuk en niet al te eenvoudig spel, waarvan je uren plezier kunt hebben. helaas wordt je elke keer als je dood bent, terug gestuurd naar de start, zodat je steeds opnieuw moet beginnen.

BLUE THUNDER

Niet wat je zou verwachten, het spel met de fraaie gevechtshelicopter uit de gelijknamige televisie-serie. het betreft hier een eenvoudiger helikopter, waarmee je een gevecht aangaat op zee en op het land. Nadat je bent opgestegen vanaf een oorlogsschip en op een bedreven wijze met je Jetcopter door alle bombardementen van electrische stormen, grond- en zeeprojectielen begint de ellende pas goed. Een nieuwe aanval van gewapende ballonnen doorkruisen je doel. Ben je er dan? Nee, opnieuw wordt je aangevallen door dodelijke Jetfighters, die uit het niets verschijnen. Zij hebben slechts een doel, jouw vernietigen. Alleen Hyperlasers en jouw bedrevenheid kunnen je levend houden en je onmogelijke opdracht completeren.

Als de enige overlevende probeer je achter de vijandelijke linies in te dringen. Van één van de eilanden, welke zwaar wordt beschermd door een complex defensie scherm, moet je je kameraden proberen te redden en ze terugvliegen naar je vloot. Maar voor je dat kunt doen moet je eerst een nucleaire reactor vernietigen. Niettemin een spel, afgezien van de graphics, waarbij je behendig te werk moet gaan. Steeds weer kom je voor verrassingen te staan. Telkens wordt je weer teruggezet naar het dek van je schip. Na elke succesvolle missie wordt je opnieuw gebriefed. Naast dit alles dien je de brandstof van je Jetcopter in de gaten te houden. Een onderhoudend spel met veel afwisseling voor wat betreft de scenario's.

ROGUE

Van Mastertronic kregen we het spel Rogue in handen. Een game waarbij moet zoeken in de Dundgeons of Doom naar het amulet van Yndor. Je enige alternatief is je leven. Voor bescherming heb je enige wapens ter beschikking, zoals je vertrouwde scepter en een boog met een aantal pijlen. Als voedsel heb je genoeg voor slechts één maaltijd. Als je de grot eenmaal binnen bent zul je meer voedsel vinden en betere wapens, die eerdere onfortuinlijke zoekers zijn achtergelaten. Onderweg tref je allerlei verborgen dingen aan, zoals goudstukken, magische zaken en ringen die je verder helpen op zoek naar het amulet. Maar tussen jou en je doel ontmoet je vervaarlijke monsters en duivelse vallen die je zullen testen op je vaardigheden. Als je slaagt in je missie zal je worden bijgeschreven in de GuildMasters of Fame, als je faalt dan heb je in ieder geval waardevolle gegevens voor een volgende poging om het amulet te vinden. Nadat je het spel hebt geladen, wordt je naam gevraagd. Een menu verschijnt met enkele keuzes voor wat betreft het selecteren van een joystick, alsmede het inladen van een eerder gesaved spel. Verder worden onderin het scherm een aantal gegevens weergegeven, zoals het aantal keren dat je nog geraakt mag worden voor je wordt gedood, de fysieke sterkte (hoe hoger, hoe beter) en de beschermende waarde van de wapens die je bij je draagt. Het scherm geeft grotendeels de grot aan van het level waarin je speelt. Vier commando's staan je ter beschikking om te zoeken, te rusten en om op en neer te gaan. Het is niet mogelijk om twee wapens tegelijk bij je te dragen. De objecten worden opgepikt als je er overheen loopt. Een leuk spel met veel variaties. Een adventure met enige allure, zeker als je het kwa prijs gaat bekijken. Voor een bedrag van om en nabij de 15 gulden een aardig en slim doordacht concept. De graphics zijn echter niet van zo'n verrassende kwaliteit.

Bert Venema

Koala painter gekraakt

Tover uw eigen tekeningen op het scherm



Er zijn voor de Commodore 64 verschillende tekenprogramma's te koop waarmee men, met een beetje talent, mooie kleurentekeningen kan maken. Een nadeel van die tekenpakketen is echter dat er niet verteld wordt hoe de amateur-programmeur die schilderijen in zijn eigengemaakte programma's op een gebruikersvriendelijke manier kan verwerken. Het volgende programma echter zorgt er voor dat men met een simpele commando de eigengemaakte (in dit geval met KOALA PAINTER) tekeningen op het scherm kan toveren.

Het programma

Het programma zelf is geschreven in machinetaal, maar kan door een simpele BASIC-listing in het geheugen geladen worden. De KOALA PAINTER-tekeningen dienen wel op een diskette te staan.
Type nu eerst listing 1 in (zie kader onder aan de pagina). Het programma heeft een ingebouwde CHECKSUM dus fouten in de DATA-regels kunnen makkelijk opgespoord worden.
Vervolgens RUN je het programma. Het machinetaal-programma staat nu in het

```
2000 FORL=0TO14:CX=0:FORD=0TO15:READA:CX=CX+A:POKE49152+L*16+D,A:NEXTD
2010 READA:IFA<>CXTHENPRINT"ERROR IN LINE";2040+(L*10):STOP
2020 NEXTL:END
2040 DATA76,49,192,76,14,192,76,121,192,234,234,234,234,234,169,6,2333
2050 DATA141,33,208,173,24,208,41,247,141,24,208,173,17,208,41,223,2110
2060 DATA141,17,208,173,22,208,41,239,141,22,208,32,68,229,96,234,2079
2070 DATA234,32,253,174,162,15,169,63,157,239,192,202,208,250,32,158,2540
2080 DATA173,32,143,173,160,0,177,100,133,251,200,177,100,133,252,200,2404
2090 DATA177,100,133,253,160,0,177,252,153,247,192,200,196,251,208,246,2945
2100 DATA234,234,162,8,160,1,32,186,255,169,15,162,240,160,192,32,2242
2110 DATA189,255,169,0,32,213,255,234,234,169,0,160,96,133,95,132,2366
2120 DATA96,169,64,160,127,133,90,132,91,169,64,160,63,133,88,132,1871
2130 DATA89,32,191,163,169,64,160,127,133,95,132,96,169,40,160,131,1951
2140 DATA133,90,132,91,169,232,160,7,133,88,132,89,32,191,163,169,2011
2150 DATA40,160,131,133,95,132,96,169,16,160,135,133,90,132,91,169,1882
2160 DATA232,160,219,133,88,132,89,32,191,163,234,234,169,1,141,33,2251
2170 DATA208,173,24,208,9,8,141,24,208,173,17,208,9,32,141,17,1600
```

Listing 1

geheugen vanaf adres 49152 en de BASIC-listing heb je nu niet meer nodig dus die kun je met een NEW verwijderen.

De commando's

Met de volgende drie commando's kun je in je BASIC-programma de diverse machinetaal-onderdelen oproepen. Vooraf moet je wel je diskette met daarop de diverse KOALA PAINTER-tekeningen in je disk drive plaatsen.

```
SYS 49152,"filenaam"
```

De tekening wordt nu vanaf disk geladen en op het scherm geplaatst. 'Filenaam' is natuurlijk de naam van je tekening (dezelfde naam die je gebruikt hebt om hem via KOALA PAINTER te 'saven').

```
SYS 49155
```

Het scherm wordt gewist en er wordt teruggekeerd naar tekst-mode.

```
SYS 49158
```

Mocht de tekening nog in het geheugen zijn dan wordt die weer op het scherm gebracht. De tekening hoeft dan niet weer opnieuw geladen te worden.
Het is misschien makkelijker om adres 49152 in een variabele op te slaan:

```
10 V=49152
20 SYS V,"filenaam":REM
   SCHERM AAN
30 SYS V+3:REM SCHERM UIT
40 SYS V+6:REM SCHERM TERUG
```

De volgende listing laat het resultaat nog eens zien. Type wel inplaats van 'filenaam' de gewenste tekeningnaam in en stop de juiste diskette in je drive.

Toepassingen

Je kunt natuurlijk op ieder moment zoveel tekeningen in het geheugen laden als je wilt. De oude tekening wordt namelijk simpelweg vervangen door de nieuwe. Het aantal gebruikte geheugen blijft dus hetzelfde. Ook hoeft je tekening niet in dataregels gezet te worden.

```
 10 REM *** TEKENING LADEN
    ***
 20 SYS 49152,"FILENAAM"
 30 FOR T=0 TO 3000:NEXT T
 40 REM *** SCHERM UIT
    *******
 50 SYS 49152+3
 60 PRINT"TERUG NAAR TEKST"
 70 FOR T=0 TO 3000:NEXT T
 80 REM *** SCHERM AAN
    *******
90 SYS 49152+6
100 GOTO 100
```

Listing 2

LISTING 3

```
****************sprongadressen
., C000   4C 31 C0 JMP $C031
., C003   4C 0E C0 JMP $C00E
., C006   4C 79 C0 JMP $C079
., C009   EA       NOP
., C00A   EA       NOP
., C00B   EA       NOP
., C00C   EA       NOP
., C00D   EA       NOP
***DEEL A**********scherm uit
., C00E   A9 06    LDA #$06
., C010   8D 21 D0 STA $D021
., C013   AD 18 D0 LDA $D018
., C016   29 F7    AND #$F7
., C018   8D 18 D0 STA $D018
., C01B   AD 11 D0 LDA $D011
., C01E   29 DF    AND #$DF
., C020   8D 11 D0 STA $D011
., C023   AD 16 D0 LDA $D016
., C026   29 EF    AND #$EF
., C028   8D 16 D0 STA $D016
., C02B   20 44 E5 JSR $E544
   ;clear screen
., C02E   60       RTS
., C02F   EA       NOP
., C030   EA       NOP
***DEEL B******filenaam halen
., C031   20 FD AE JSR $AEFD
   ;komma?
., C034   A2 0F    LDX #$0F
., C036   A9 3F    LDA #$3F
., C038   9D EF C0 STA $C0EF,X
., C03B   CA       DEX
., C03C   D0 FA    BNE $C038
., C03E   20 9E AD JSR $AD9E
   ;filenaam halen
., C041   20 8F AD JSR $AD8F
   ;string?
., C044   A0 00    LDY #$00
., C046   B1 64    LDA ($64),Y
., C048   85 FB    STA $FB
., C04A   C8       INY
., C04B   B1 64    LDA ($64),Y
., C04D   85 FC    STA $FC
., C04F   C8       INY
., C050   B1 64    LDA ($64),Y
., C052   85 FD    STA $FD
., C054   A0 00    LDY #$00
., C056   B1 FC    LDA ($FC),Y
., C058   99 F7 C0 STA $C0F7,Y
., C05B   C8       INY
., C05C   C4 FB    CPY $FB
., C05E   D0 F6    BNE $C056
., C060   EA       NOP
., C061   EA       NOP
***DEEL C***************laden
., C062   A2 08    LDX #$08
., C064   A0 01    LDY #$01
., C066   20 BA FF JSR $FFBA
   ;disk
., C069   A9 0F    LDA #$0F
., C06B   A2 F0    LDX #$F0
., C06D   A0 C0    LDY #$C0
., C06F   20 BD FF JSR $FFBD
   ;filenaam
., C072   A9 00    LDA #$00
., C074   20 D5 FF JSR $FFD5
   ;laden
., C077   EA       NOP
., C078   EA       NOP
***DEEL D***********copieeren
., C079   A9 00    LDA #$00
., C07B   A0 60    LDY #$60
., C07D   85 5F    STA $5F
., C07F   84 60    STY $60
., C081   A9 40    LDA #$40
., C083   A0 7F    LDY #$7F
., C085   85 5A    STA $5A
., C087   84 5B    STY $5B
., C089   A9 40    LDA #$40
., C08B   A0 3F    LDY #$3F
., C08D   85 58    STA $58
., C08F   84 59    STY $59
., C091   20 BF A3 JSR $A3BF
   ;tekening verplaatsen
., C094   A9 40    LDA #$40
., C096   A0 7F    LDY #$7F
., C098   85 5F    STA $5F
., C09A   84 60    STY $60
., C09C   A9 28    LDA #$28
., C09E   A0 83    LDY #$83
., C0A0   85 5A    STA $5A
., C0A2   84 5B    STY $5B
., C0A4   A9 E8    LDA #$E8
., C0A6   A0 07    LDY #$07
., C0A8   85 58    STA $58
., C0AA   84 59    STY $59
., C0AC   20 BF A3 JSR $A3BF
   ;kleuren verplaatsen
., C0AF   A9 28    LDA #$28
., C0B1   A0 83    LDY #$83
., C0B3   85 5F    STA $5F
., C0B5   84 60    STY $60
., C0B7   A9 10    LDA #$10
., C0B9   A0 87    LDY #$87
., C0BB   85 5A    STA $5A
., C0BD   84 5B    STY $5B
., C0BF   A9 E8    LDA #$E8
., C0C1   A0 DB    LDY #$DB
., C0C3   85 58    STA $58
., C0C5   84 59    STY $59
., C0C7   20 BF A3 JSR $A3BF
   ;kleuren verplaatsen
., C0CA   EA       NOP
., C0CB   EA       NOP
***DEEL E*********scherm aan
., C0CC   A9 01    LDA #$01
., C0CE   8D 21 D0 STA $D021
., C0D1   AD 18 D0 LDA $D018
., C0D4   09 08    ORA #$08
., C0D6   8D 18 D0 STA $D018
., C0D9   AD 11 D0 LDA $D011
., C0DC   09 20    ORA #$20
., C0DE   8D 11 D0 STA $D011
., C0E1   AD 16 D0 LDA $D016
., C0E4   09 10    ORA #$10
., C0E6   8D 16 D0 STA $D016
., C0E9   60       RTS
```

Listing 3

Dat maakt het programma handig voor b.v. zelfgeschreven adventures. De meeste commerciële adventures-programma's maken gebruik van tekeningen die al in het programma zelf zitten. Dat maakt het aantal tekeningen natuurlijk beperkt. Door echter je vele tekeningen op een aparte schijf te zetten kun je nu op ieder gewenst moment de juiste tekening laden. Je kan natuurlijk ook meer serieuse programma's van graphics voorzien.

De listing

Voor de machinetaal-programmeurs volgt nu de listing van het programma.

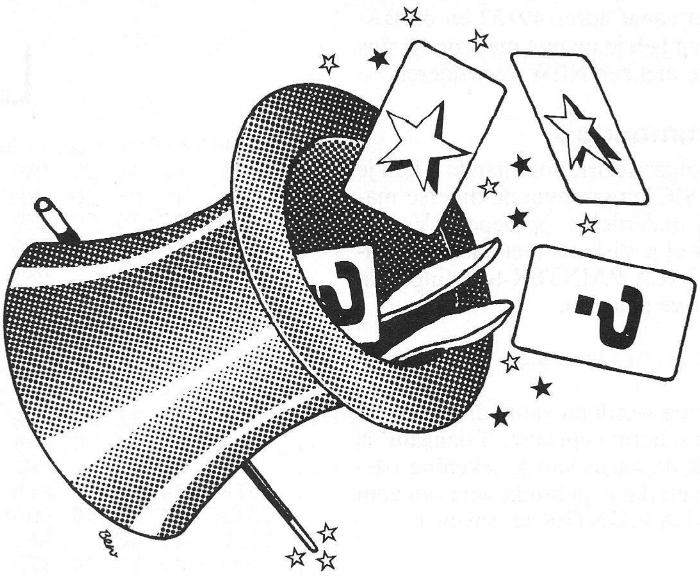
De werking

Ik heb het programma in diverse delen gesplitst. Hier volgt in het kort een beschrijving van de werking.

Deel A:
Hier wordt op de normale manier het scherm uitgeschakeld. Voor degene die niet bekend zijn met de werking van BIT-MAP-MODE, zie de PROGRAMMER'S REFERENCE GUIDE. 'JSR $E544' is een KERNAL-routine en de machinetaal-versie van CLR/HOME. Het wist dus het scherm (die staat namelijk vol met kleurencodes).

Deel B:
De ingevoerde filenaam wordt binnengehaald. In de inhoudsopgave zie je behalve de filenaam nog meer tekens die KOALA PAINTER zelf gebruikt om de diverse tekeningen uit elkaar te houden. Deze tekens hebben wij niet nodig. D.m.v. de 'joker', het vraagteken, slaan wij die tekens over. De file wordt dus als b.v. "SEARCHING FOR ???????NAAM???" geladen. De zo veranderde filenaam wordt vanaf adres $C0F0 opgeslagen.

Deel C:
Het laden van de file zelf. Achtereenvolgens wordt het devicenummer (8), het secundaire devicenummer (1), de lengte van de filenaam (15 incl. vraagtekens) en het adres van de filenaam ($C0F0) gegeven.



Deel D:
De KOALA PAINTER-tekening is als volgt opgeslagen:

```
24576-32575 : Tekening
32576-33575 : Multi-colours1+2
33476-34575 : Multi-colour 3
```

Wij gebruiken het echter als volgt.

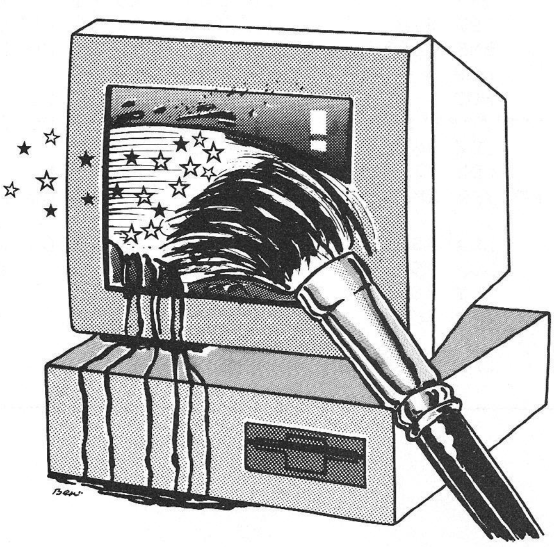
```
8192-16191  : Tekening
1024-2023   : Multi-colours 1+2
55296-56295 : Multi-colour 3
```

Deze routine kopieert de diverse delen op de juiste plaats. Ik gebruikte hiervoor een routine uit BASIC-ROM. Laten we de werking daarvan een bekijken met als voorbeeld $C079-$C091

```
., C079   A9 00      LDA #$00
., C07B   A0 60      LDY #$60
., C07D   85 5F      STA $5F
., C07F   84 60      STY $60
., C081   A9 40      LDA #$40
., C083   A0 7F      LDY #$7F
., C085   85 5A      STA $5A
., C087   84 5B      STY $5B
., C089   A9 40      LDA #$40
., C08B   A0 3F      LDY #$3F
., C08D   85 58      STA $58
., C08F   84 59      STY $59
., C091   20 BF A3   JSR $A3BF
```

Eerst wordt adres $6000, het oude blok geheugen in $5F/$60 opgeslagen. Dan het eindadres+1 van het oude blok in $5A/$5B en dan het eindadres+1 van het NIEUWE blok in $58/$59. Tenslotte wordt d.m.v. 'JSR A3BF' de verschuifroutine aangeroepen. Het resultaat is dat alles van $6000-$7F3F gecopieerd wordt naar een blok geheugen met als eindadres $3F3F. Het beginadres van dat nieuwe blok moet je dus zelf berekenen.

Deel E:
Het Bitmap-scherm wordt ingeschakeld en de tekening is te zien.



Verdere voorstellen

De tekening is nu opgeslagen vanaf adres 8192, maar dat kan makkelijk verschoven worden naar een hogere geheugenplaats zodat je meer geheugen voor je eigen programma overhoudt. Nu je weet waar je tekening zich bevind kan je er natuurlijk nog veel meer mee doen. Je zou b.v. een programma kunnen schrijven om je tekening af te drukken op je printer. Ook zou je je tekening als SPRITE-editor kunnen gebruiken door een tekening te vullen met kleine tekeningen van 24x21 grootte. Door dan die tekeningen te copieeren naar je SPRITE-adressen kun je b.v. een wandelend voetbalpoppetje samenstellen.

Succes ermee

Rene Janssen

Basis Basic (25)

Basic van A tot Z.

Na een lange periode van onderbreking gaan verder met onze uitleg van de diverse Basic-opdrachten en functies in onze Basic-cursus. Op dit moment zijn we bezig om in alfabetische volgorde de meeste, algemene commando's in Basic aan een nader onderzoek te onderwerpen. Wat hier door Jan Bodzinga wordt behandeld is in grote lijnen niet alleen te gebruiken op de Commodore-reeks, maar kan ook worden toegepast in iedere andere computer die met Basic overweg kan, al is de basis wel bedoeld voor de Commodore *pur sang*.

oewel we er even tussenuit zijn geweest met deze serie, pakken we nu de draad weer op waar we gebleven zijn. Zoals de trouwe lezers ongetwijfeld zullen weten, hebben we in de laatste aflevering een begin gemaakt met een alfabetische reis langs alle Basic -opdrachten en -functies. We zijn op dit moment nog niet al te ver, want na de functies ABS, AND en ASC gaan we nu verder met een trigometrische functie : ATN.

ATN

Type : BASIC functie, reken (wis) kundig
Doel : Het berekenen van de arctangens van het argument dat tussen de haken aan de functie ATN wordt meegegeven.
Dit argument kan een wiskundige expressie zijn, zowel positief als negatief.
De hoek die door ATN wordt gemeten, is de hoek van een rechthoekige driehoek, die ligt tegenover de zijde waarvan de lengte door het argument wordt bepaald.

In het bovenstaande voorbeeld wordt dus de linkerhoek door ATN berekend, waarbij de rechterzijde (X) zorgt voor het argument.
De uitkomst van deze functie wordt gegeven in radialen.

Mode : ATN kan zowel in de programmamode als rechtstreeks vanaf het toetsenbord als functie worden ingetoetst.

Syntax : ATN(rekenkundige expressie)
De expressie tussen de haken moet uiteindelijk te herleiden zijn tot een numerieke waarde. Een string of foute expressie heeft een foutmelding tot gevolg, zoals 'SYNTAX ERROR', 'TYPE MISMATCH' of 'DIVISION BY ZERO ERROR'. Als de uitkomst van de expressie te groot is, dan ook laat de Commodore het afweten met een 'OVERFLOW ERROR'. De maximale waarde die tussen de haken kan worden gezet, heeft te maken met het grootste getal dat door de floating point accumulator van de computer kan worden verwerkt en ligt ongeveer rond de 1.7E38. Een fors getal dus, ruim vertaald 17 met 37 nullen erachter. Iets in de miljarden waarvan de schrijfwijze beter voor te stellen valt dan de uitspraak.
De ATN()-functie kan worden gebruikt aan de rechterkant van een statement waarin waarden aan variabelen worden toegekend (assignments) ofwel na een logische expressie als bijvoorbeeld IF. Ook na een PRINT-opdracht kunnen we de functie ATN() tegenkomen.

Gebruik

Het zal voor de normale computer-programmeur niet erg vaak voorkomen, dat de functie ATN() wordt gebruikt. Hoewel, voor zover ik heb kunnen nagaan, al vanaf de eerste versies Basic in 1962/65 deze functie tot de standaardset behoort wordt er weinig gebruik van gemaakt. Ik moet dan ook eerlijk bekennen, dat voor mij het gebruik van deze ArcTaNgens ook niet zo vaak -zeg maar: nooit- voorkomt. Maar de mogelijkheid bestaat, en als je wat dieper doordenkt is het helemaal niet zo gek, om wat met deze functie te stoeien. Zeker de fervente wiskundigen onder ons zullen zich hier wel sterk voor willen maken.
Om het gebruik van de ATN-functie wat toe te lichten hierbij een paar voorbeelden:

```
I     PRINT ATN(0)
```

Hierbij wordt de arctangens gevraagd van 0, ofwel, gezien ons voorbeeld van een lijn met lengte 0, dus niet bestaand. De uitkomst hiervan is dan ook natuurlijk 0 radialen.

Ia PRINT ATN(20)

De uitkomst van deze bewerking is 1.520838 rad. en deze uitkomst ligt ergins in het midden van de mogelijke range. Zoals de meesten en zeker de wiskunstenaars onder ons wel zullen weten, kan de uitkomst (in radialen) alleen liggen tussen -pi/2 en +pi/2 radialen. Dit komt, omdat er in het geval van de arctangens nooit sprake kan zijn van een hoek groter dan +/- 90 graden. Met 'pi' wordt de constante bedoeld, die een waarde heeft van ongeveer 3.141592 met daarachter nog een hele serie decimalen.

Ib PRINT ATN(-1.7E38)



Door de uitkomst van ATN met dit argument (1.7E38) zien we, dat we hiermee de maximale waarde voor ATN hebben bereikt, dus 'oneindig' als argument ofwel *-1.570796* in radialen. We zitten dan ook heel dicht bij de hoek van 90 graden. We zien dit direct, als we de driehoek in het voorbeeld bekijken. Hoe langer we de lijn X maken, hoe groter de hoek ertegenover wordt, waarvan de arctangens berekend wordt. De hoek kan echter nooit groter worden dan een (bijna) rechte hoek, en zelfs dan is de lengte van lijn X al tot het oneindige opgelopen. Omgezet in radialen komen we daardoor op pi/2. Gaan we nog verder, door bijvoorbeeld 1.8E38 als argument aan ATN mee te geven, dan retourneert de Commodore een 'OVERFLOW ERROR'. Dit komt echter niet zozeer van de ATN-functie, die loopt tot in het oneindige verder, maar door de bovengrens van de mogelijke getallen die met een computer kunnen worden verwerkt.

```
II   Y= ATN(X) * 180/pi
```

Deze ATN-functie laat zien, hoe de radiale uitkomst kan worden omgezet in graden. Voor de waarde 1.570796 van X krijgen we voor Y de waarde 89.9979, waardoor we al erg dicht bij een rechte hoek uitkomen. De uitkomst van deze rekenarij is echter al degelijk gekleurd door de afrondingen die de computer er wel op los moet laten, maar de principes lijken me duidelijk.

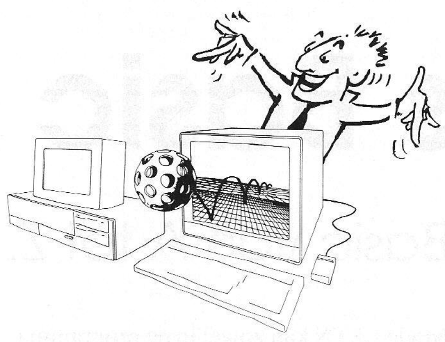
```
III  XX = -ATN(YV/ZV)
     YY = -ATN(XV/ZV)
```

Op de manier van voorbeeld III kan men aan de hand van coördinaten YV en XV en de derde dimensie coördinaat ZV een punt in de ruimte berekenen en bijvoorbeeld naar een plotter sturen, of gebruiken in het werken met CAD-applicaties. Veel van dit soort moderne software maakt dan ook wel degelijk gebruik van de arctangens, al zal dit niet primair in Basic gebeuren.
Het laatste voorbeeld komt uit een boek over het berekenen van kompaskoersen en afstanden:

```
IV   AR = ATN(
     (W1-W2)/(N1-N2) )
```

W1,W2,N1,N2 zijn hierbij coördinaten van twee verschillende kompasrichtingen, waardoor met hulp van de ATN-functie de juiste afstand en richting kan worden berekend.

Al met al toch nogal wat mogelijkheden, al liggen de meeste van deze niet zo erg voor de hand. Niettemin goed om ook de



ATN-functie ergens in je achterhoofd te houden, voor het geval een dergelijke berekening nog eens nodig is.
Als laatste wil ik hierbij nog opmerken, dat ook bij statistici de ATN nog wel in ere wordt gehouden, omdat in een bijzonder kleine omvang, tussen -1.57 en +1.57 het met hulp van de ATN-functie mogelijk is geworden de complete getallenreeks uit te drukken. Zodoende kun je beschikken over een zeer geconcentreerde wijze van het opslaan van getallen, al is natuurlijk de indicatie nogal grof. De door ATN berekende waarden kunnen dan uiteraard weer worden teruggezet naar de waarde van hun oorspronkelijk argument.

Werkwijze

De manier waarop de computer de arctangens berekent, is nogal ingewikkeld en gaat uit van een serie van 12 in het ROM-geheugen van de Basic interpreter aanwezige constanten.
Aan de hand van diverse berekeningen tussen deze 12 vaste getallen en het aan ATN meegegeven argument wordt uiteindelijk de arctangens berekend. De waarde van ATN ligt tussen Pi/2 en -Pi/2 (1.57) en wordt uitgedrukt in radialen.
Het omzetten van radialen naar graden kan door de uitkomst van ATN te delen door het getal Pi, waarna het geheel met 180 moet worden vermenigvuldigd. zie hiervoor ook voorbeeld II.

Opmerking

Als kanttekening wil ik nog opmerken, dat de ATN-functie beslist niet moet worden verward met de afkorting ATN die wordt gebruikt bij de I/O met diverse poorten, zoals de RS-232C en de IEEE bus. ATN betekent in dat geval 'ATTENTIE' en dit heeft niets te maken met de arctangens.

```
CHR$
Type: Basic functie, string
   met numeriek argument.
```

Doel

De functie CHR$() bewerkt ieder numeriek argument tussen 0 en 255 tot een string (alfa-numeriek) met een lengte van 1 karakter.
De door CHR$ gevormde string bevat het met het argument overeenkomstige teken in de CBM-ASCII tabel. Hiermee is het mogelijk bepaalde 'stuur'codes zoals de RETURN-code en aanhalingstekens te kunnen bewerken en printen, die anders onmogelijk kunnen worden toegepast.

Mode : Zowel in direkte (keyboard) als programma-mode kan de functie CHR$() worden gebruikt.

```
Syntax : CHR$(numerieke
   expressie)
```

De uitdrukking tussen de haakjes, het argument, moet uiteindelijk kunen worden geëvalueerd tot een getal dat ligt tussen 0 en 255, de beide uiterste waarden meegerekend. Als het geen integer getal is, zal het fractionele gedeelte (achter de komma) worden weggelaten, zonder dat hierbij van afronding op de wiskundige manier sprake is. Dus CHR$(-0.1) en CHR$(500) en CHR$(A3$) geven alle een foutmelding op de computer. Vooral 'ILLEGAL QUANTITY' is een veel voorkomende melding bij deze functie.

Gebruik

Zoals reeds in de definitie opgemerkt, kunnen we deze functie heel goed gebruiken om bepaalde codes, als bijvoorbeeld de carriage-return te bewerken. Met name de RETURN-code (CHR$(13)) is een goed voorbeeld, omdat we binnen Basic, zoals bijna bij alle programmeertalen, de RETURN-toets gebruiken om een regels tekst of programma in te typen, terwijl dezelfde RETURN wordt benut om het einde van een INPUT aan de computer door te geven. O[het moment dat we normaal gesproken een RETURN-code naar de printer willen sturen en we drukken hierbij tijdens het programmeren op de RETURN-toets, dan gebeurt er niets anders, dan dat de programmaregel op dat moment wordt afgesloten en de cursor op een nieuwe regel staat te knipperen. Wat we echter voor ogen hadden, zal met de niet afgemaakte programmaregel dan ook niet kunnen gebeuren.
Een paar voorbeelden kunnen dit prachtig toelichten:

```
I   PRINT#4, CHR$(13)
```

Hiermee wordt een RETURN-code naar de printer gestuurd. Daardoor wordt een nieuwe regel op papier begonnen. We kunne dezelfde manier gebruiken om op

het scherm één of meer nieuwe regels te maken :

```
Ia  +FOR I = 0 TO 12 : PRINT
    CHR$(13): NEXT
```

Dertien nieuwe, schone regels krijgen we op het scherm, door bovenstaande commando-functiereeks uit te voeren.

```
II   PRINT CHR$(7)
```

De zevende code in de ASCII-tabel wordt hiermee 'geprint'. Voor ingewijden betekent dit, dat er een bel gaat rinkelen, anderen moeten maar eens proberen wat er in de computer aan het werk wordt gezet door deze code te printen op scherm of printer.

```
III   T$ = CHR$(34) + "JARIG
    ZIJN " + CHR$(34)
```
PRINT T$

In dit voorbeeld wordt de string T$ gevormd door de tekst 'JARIG ZIJN' die in het programma al tussen " " (aanhalingstekens) staat. Om ook op de printer (of het scherm) geforceerd aanhalingstekens te kunnen printen, moeten we uit de ASCII-tabel de code zoeken voor het "-teken. Dit is ASCII-34. Door nu een string-optelling te maken, met de tekst tussen twee CHR$(34) codes, krijgen we uiteindelijk het gewenste resultaat.

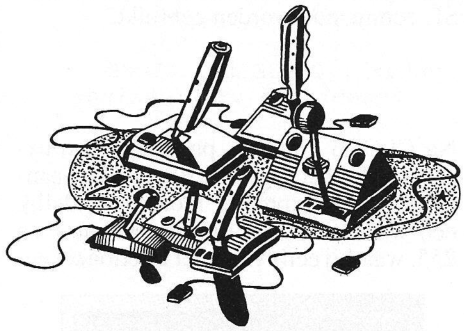
```
IV   OPEN 4,4:PRINT#4,
    CHR$(27); CHR$(65);12 :
    CLOSE 4
```

Dit voorbeeld komt uit de handleiding voor een matrixprinter, waardoor via Basic de printer wordt ingesteld met een bepaald aantal regels per inch. In dit geval wordt aan de hand van de numerieke waarde 12, het laatste argument, de regelspatiëring gezet op 12/72 van een inch, ofwel 1/6. Wijziging van dit getal geeft uiteraard effect op de printer.
CHR$(27) geeft hierbij de string-variant van de ESC-toets, terwijl CHR$(65) ook zou kunnen worden vervangen door een 'a', maar dit geeft wat problemen bij de ASCII-conversie, die -zoals bekend- bij de Commodore nogal afwijkt van de standaard, waaraan ook de meeste matrixprinters zich gelukkig houden.

```
V    N$= CHR$(32)+CHR$(32)+N$
```

De string N$ wordt door voorbeeld V gewijzigd in dezelfde string, echter voorafgegaan door twee spaties, waar dit dan ook voor nodig mag zijn. Het geheel kan natuurlijk net zo goed als volgt worden geprogrammeerd:

```
V  aN$ = "  " + N$
```



en is daardoor wel korter, maar zeker niet overzichtelijker en duidelijker geworden. Het kan zeker nuttig zijn om bepaalde strings op deze manier op te lengen, al was het maar om alle strings op dezelfde lengte te krijgen.

```
VI   FOR I = 32 TO 255 :
    PRINT CHR$(I);CHR$(32);:
    NEXT
```
Op het eerste gezicht zou je het niet zeggen, maar deze regel uit voorbeeld VI print de gehele (printbare) ASCII-tabel op het scherm. Een paar aanvullingen voor de layout en je hebt je eigen ASCII-controle op scherm of printer.
Voor de volledigheid zullen we de ASCII-codes die niet te printen zijn en toch standaard-acties veroorzaken, hierbij even op een rijtje zetten :

Waarde	Naam	Actie
03	STOP	Break in programma
07	BELL	Beep of bel in printer/computer
08	BS	Backspace (cursor terug)
09	HTAB	Horizontal TAB
10	LF	Nieuwe regel, cursor zelfde plek
11	VT	Vertikale TAB
12	FF	Nieuw blad / scherm leeg
13	CR	Carriage RETURN
14	TXT	Selectie tekst/ grafische mode
17	CD	Cursor neer
18	REV	Reverse schrift aan
19	CH	Cursor naar links boven
20	DCHR	Weghalen karakter
21	DLINE	Weghalen regel
22	EREND	Weghalen tot eind
27	ESC	Escape code
29	CR	Cursor naar rechts

Er is nog wel een serie mogelijkheden om de ASCII-codes van 0 tot 32 in te vullen, maar bovenstaande opsomming komt het dichtst bij de indertijd door Commodore gedefinieerde karakterset. Vooral bij het gebruik van niet-Commodore printers, m.n. de Hewlett-Pacard Laserjet en al zijn compatibelen kan niet zonder het gebruik van de CHR$()-functie deze apparaten naar behoren laten werken.

```
VII  FOR J = 2048 TO 3000
    PRINT CHR$(PEEK(J));
    NEXT
```

Op deze manier wordt het mogelijk een gedeelte uit het computergeheugen, in dit geval de adressen 2048 tot 3000 in een redelijk leesbare vorm op het scherm te zetten. Ook hier een goed gebruik van de functie CHR$(). Het overige van deze functie spreekt voor zichzelf.

Werkwijze : In de computer wordt de verwerking van CHR$() als volgt gedaan. Als eerst wordt de inhoud van het argument gecontroleerd op geldigheid (0-255). Vervolgens wordt in het computergeheugen ruimte voor een nieuwe string gemaakt, met een lengte van 1 byte. De enkele Byte uit de karakterrom-ASCII tabel wordt op dit stringadres gezet. Is de string aan een variabele toegekend (X$=CHR$(33)), dan worden de pointers voor deze stringnaam gecorrigeerd, zodat ze wijzen naar de stringadressen, zojuist gealloceerd door CHR$(). Anders worden de gegevens doorgegeven aan de vervolgroutines, zoals bijvoorbeeld bij de opdracht PRINT CHR$(12+33-1*X)

Opmerking

Ook bij IBM-compatible computers en andere Basic-dialecten wordt deze ftie gebruikt.
Alleen moet daarbij rekening worden gehouden met een andere opbouw van de ASCII-tabel.
Er is wat het de Commodore betreft nog een serie opmerkingen te maken die men voor een juiste behandeling van deze functie nodig heeft.
Als eerste wil ik hierbij noteren, dat de functie CHR$() het omgekeerde bewerkt van de functie ASC.
De programmaregel :

```
X = ASC(X$) : X$ = CHR$(X)
```
ofwel :
```
X = ASC(CHR$(X))
```

laten dit maar al te duidelijk zien. Daardoor kan de functie CHR$() ook goed worden gebruikt om een conversie te bewerken van schermcodes naar het printen van dezelfde karakters of vice versa.
Een fenomeen betreft de waarde NUL. *CHR$(0)* bekleedt daarom een heel speciale plek in deze functie.
Probeer maar eens *PRINT CHR$(0)*, dan zie je niets bijzonders op het scherm, maar LEN (CHR$(0)) evalueert wel degelijk tot 1.

We kunnen, hierop voortbordurend een aardig, slimme en snelle beveiliging op touw zetten, want ook de regel :

```
X$ = "123"+CHR$(0)+"456"
```

print als *"123456"* maar bij het opvragen van *LEN(X$)* krijgen we wel degelijk 7 als resultaat. Iets wat bijna niemand in de gaten zal hebben, als daar wat mee wordt gegoocheld. Ook een functie als *VAL(X$)* komt in dat geval niet verder dan een antwoord 123 en dat verwacht niemand. Probeer maar eens en je zult verbaasd zijn over het resultaat en de moeite die iedereen krijgt om een goed opgezette routine op deze wijze te kunnen kraken.
Een laatste opmerking heeft ook betrekking op CHR$(0), maar dan in de vorm van een vergelijking. Naar de mens gesproken zou CHR$(0) identiek moeten zijn met de nulstring ofwel "", maar een eenvoudige vergelijking PRINT CHR$(0) > "" evalueert tot FALSE, iets wat niet geheel wordt verwacht.

```
CLOSE
Type :Basic commando,
    Input/Output
```

Doel

Met CLOSE wordt de dataoverdracht via één van de input/output kanalen van de computer afgesloten. Tegelijk wordt het bestand en zijn attributen verwijderd uit de file-I/O-tabellen in het computergeheugen. Met files die uitsluitend betrekking hebben op toetsenbord en scherm, wordt door het uitvoeren van de CLOSE-opdracht geen verdere actie ondernomen. Cassette-bestanden die uitsluitend open waren in de READ-(lees-)modus ondergaan hetzelfde lot.
Bij files open in READ/WRITE (lees/schrijf-)modus wordt voor het sluiten nog een NUL-byte weggeschreven, om daarmee het eind van het file aan te geven.
Bij een open (cassette)bestand, waarvan het secundaire adres de waarde 2 heeft, wordt bij CLOSE ook een 'Tape-header' op de cassette weggezet en tevens een END-OF-TAPE markering op de cassette aangebracht.
Bij het gebruik van CLOSE in samenhang met CBM-hardware, als bijvoorbeeld de 1541 diskdrive, wordt aan de hand van deze opdracht een END-OF-FILE markering meegeschreven in het laatste blok van het bestand op de diskette. Daardoor wordt het later teruglezen van de ketting van blokken waaruit dit file bestaat, op het juiste moment gestopt. Bij niet-Commodore hardware dient het CLOSE commando door de specifieke hardware op een eingen manier te worden geïnterpreteerd en uitgevoerd.

Mode : Zowel in programmamode, als direkt vanaf het toetsenbord kan het CLOSE-commando worden gebruikt.

```
Syntax : CLOSE LF; CLOSE
    >numerieke uitdrukking@
```

Na CLOSE komt als parameter een expressie die het logische filenummer aanduidt. Deze expressie moet uiteindelijk een waarde hebben die ligt tussen 1 en 255, waarbij een eventueel fractioneel ge-



deelte in de expressie naar beneden wordt afgerond. Er volgt bij een niet bestaand file geen foutmelding. CLOSE gebruikt bij de verwerking alleen de eerste parameter. Eventueel volgende parameters worden door CLOSE niet onderkend en in het programma tijdens de uitvoer domweg genegeerd.

Gebruik

Het werken met CLOSE is tamelijk eenvoudig en zeker recht voor z'n raap.
In principe komt het erop neer, dat een bestand naar disk, tape of printer eerst met een OPEN-opdracht moet worden geopend. Daarmee wordt (door de programmeur) een logisch filenummer aan het bestand toegekend. Aan de hand van dit nummer blijven de diverse lees- en schrijfopdrachten ook voor het programma en de Basic-interpreter herkenbaar als behorend bij het (open) bestand met het genoemde LF-(logical file) nummer.
Als er geen actie meer hoeft te worden uitgevoerd met een bepaald open bestand, dan wordt de ruimte die dit bestand inneemt in het computergeheugen vrijgemaakt voor andere doeleinden en nieuwe bestanden door het CLOSE-commando uit te voeren.
Bovendien zorgt de uitvoer van de CLOSE-opdracht, dat eventueel nog niet verwerkte buffers van het te sluiten bestand voor de afwerking nog naar de juiste plek op disk, tape of printer worden weggezet, zodat geen gegevens verloren gaan. Bij het weglaten van CLOSE kan het daardoor heel goed gebeuren, dat een bestand onvolledig op de disk staat weggeschreven, omdat na de laatste schrijfopdracht de computer bijvoorbeeld werd uitgezet, voordat een CLOSE-commando kon worden uitgevoerd. Daardoor was geen gelegenheid meer om bijvoorbeeld de disk-directory naar behoren bij te werken, waardoor deze onvolledigheid kan ontstaan. Dergelijke bestanden krijgen in veel gevallen een '*' mee tijdens de listing van de directory, maar dit blijkt niet in alle commodore-gevallen op te gaan.

```
I   OPEN 1,1,1, "FILE":
    PRINT #1,"Dit is TEKST" :
    CLOSE 4
```

Op bovenstaande wijze wordt een volledig bericht "Dit is TEKST" naar een cassettebestand gestuurd. Hoe het werken met OPEN en PRINT# in z'n werk gaat, zullen we uitvoerig behandelen bij de betreffende commando's, vooralsnog lijkt het me voldoende te vermelden, dat met het OPEN-commando een bestand met (in dit geval) LF-nummer 1 naar de cassette (device 1) wordt geopend. Zoals je kunt raden staat de 3e parameter (1) voor het secundaire adres van dit bestand. Na de opening wordt via PRINT#LF = PRINT#1 een tekst "Dit is TEKST" naar de tape gestuurd. Het geheel wordt als laatste afgesloten door de CLOSE-opdracht.

II CLOSE 1,2,"RERERE",3,12

Door een bepaalde speling in de Basic-interpreter heeft deze opdrachtregel een identiek effect als het bovenstaande CLOSE 1. Alle parameters en tekst achter de LF-parameter wordt volledig door de computer genegeerd tot aan de eerstvolgende opdracht-delimiter, hetzij een nieuwe regel of een dubbele punt.

```
III    PRINT#2,CHR$(13):CLOSE 2
```

Hoewel al weer bijna antiek, was dit de manier waarop we een carriage return moesten genereren bij bestanden die met de Commodore Basic lager dan versie 4.0 werden gemaakt. Het eenvoudige PRINT#4 werkte daarbij niet. Een weet voor iedereen die nog zo'n oud machien bezit, of wellicht te maken krijgt met heel antiek programma's waarin deze regels maar al te vaak moesten worden gebruikt.

```
IV    OPEN 4,4 : PRINT #4,
    "BERICHT" : CLOSE 4
```

Op deze manier wordt een tekst naar de printer verstuurd.
Verder lijkt me hier geen woord Spaans bij, zeker als we dit voorbeeld vergelijken met voorbeeld I, waar een identieke pro-

cedure wordt gevolgd voor een cassettebestand. Het enige verschil is het ontbreken van een secundair adres. Dit is het geval, omdat de printer alleen een 'luisterend' apparaat is, dat zelf weinig meer aan gegevens naar de computer terugstuurt dan op gezette tijden een statusbyte, waaraan te zien valt hoe het met de printer ervoor staat. De primaire functie van een secundair adres richt zich juist op devices die veel kunnen retourneren. Maar daarover meer bij het OPEN-commando.

```
V    PRINT#4, CLOSE 4
```

Op deze wijze dient een file te worden afgesloten, waarmee in de CMD-modus is gecommuniceerd. Een belangrijke bug in het Commodore-programma. Erin geslopen zo rond 1978 en nog immer in stand gehouden, zover ik weet. Het uitvoeren van een CLOSE-opdracht laat bij de status CMD-mode het apparaat (printer of diskdrive) in een luisterende status achter, waardoor ook NA de CLOSE opdracht nogal wat gegevens verloren kunnen gaan. Vandaar dat met PRINT#4 eerst de opdracht CMD4 moet worden opgeheven, waarna het bestand op de gebruikelijke manier kan worden afgesloten.

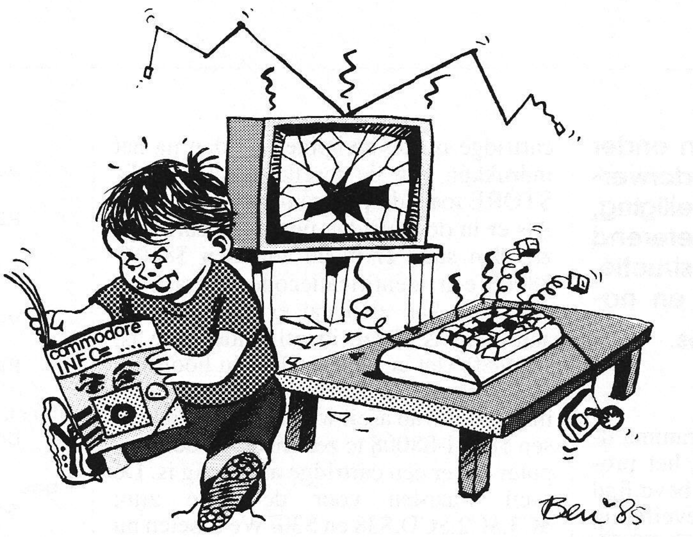
Werkwijze

Bij het ontmoeten van CLOSE in een programma wordt door de computer eerst de volgende LF-parameter ingelezen. Aan de hand van dit nummer worden de tabelling van de OPEN bestanden in RAM nagelopen, tot het juiste nummer is gevonden. Dit nummer wordt, samen met de nodige attributen in de RAM-tabellen gewoon overschreven door de gegevens van het allerlaatste OPEN bestand in RAM. Daardoor komt aan de onderzijde in de tabel ruimte vrij, zodat de maximaal 10 OPEN-bestanden op een goede manier kunnen worden benut.
Het device nummer bepaalt vervolgens of -en zo ja, meer actie moet worden ondernomen. Bij keyboard en scherm is de computer snel gereed. Voor IEEE- en RS-232- apparaten moet ook nog het kanaal worden schoongeveegd; dit gebeurt door de ROM-opdracht 'CLEAR CHANNEL', een oude getrouwe uit de tijd van de IEEE interfacing.

Opmerking

CLOSE is een KERNEL-opdracht en kan daarom op eenvoudige wijze worden aangepast aan de verschillende Basic-versies.

Als een bestand onverhoopt niet kon worden afgesloten via CLOSE, omdat voortijds een STOP/BREAK of Syntax Error is uitgevoerd of opgetreden is het toch mogelijk een 'echte' CLOSE opdracht te genereren, waardoor het bestand nog op een juiste wijze kan worden weggezet. Dit geldt natuurlijk alleen voor bestand en die ook voor WRITE zijn geopend. Bestanden waarvan alleen wordt gelezen, zijn in feite inactief op het medium, dat wil zeggen hun directe status verandert



niet, maar wordt alleen gelezen. Bij een schrijfactie verandert wel veel in een bestand, daarom is het een goede zaak, te zorgen dat het bestand hoe dan ook wordt geCLOSED.
Bij het optreden van fouten kan men als eerste proberen een CLOSE LF-opdracht in de directe mode in te toetsen, mist het logische filenummer van het betrokken bestand bekend is.
In latere versies van Commodore's Basic kan men daarvoor ook de opdracht DCLOSE gebruiken, die sowieso alle open bestanden afsluit, wat echter ook niet altijd de bedoeling is.
Een ander redmiddel bij foute nog open bestanden kan de volgende opdracht-sequens zijn :

```
OPEN 15,8,15 : CLOSE 15.
```

Hiermee wordt het ERROR-kanaal van de disk geopend, bekenen en weer gesloten. Daarmee kan het zo zijn, dat ook nog 'pending' data wordt afgewerkt op de juiste manier, maar -hoewel deze oplossing uit een Commodore-handboek afkomstig is- heb ik niet zoveel vertrouwen in deze redding.

Voor cassettebestanden is niet zoveel hoop meer, hoewel deze, als ze nooit zijn gesloten met CLOSE meestal nog wel opnieuw zijn in te lezen.
Het enige wat dan gebeurt, is dat de cassetterecorder ook na het lezen van de laatste byte gewoon door blijft lopen. Proefondervindelijk is dit punt op de tape wel te benaderen. Stop gewoon de tape, controleer in het programma of alle gegevens die op het bestand horen te staan zijn ingelezen, monteer een nieuwe tape en schrijf het gehele bestand op de juiste manier terug naar de schone tape, waarbij dan uiteraard wel een CLOSE-opdracht moet worden uitgevoerd.
Een laatste woord specifiek over disk-bestanden. Bij het werken met de diskdrive en bestanden op de diskette, kennen we twee soorten dataopslag. SEQuentiële en RELationele data. Wat betreft het CLOSE-commando ligt hierin geen verschil.
Zoals bekend worden de data van deze bestanden per blok van 254 bytes op de disk gezet. Samen met twee bytes voor het adres van het volgende blok in de ketting krijgen we hierdoor precies de 256 bytes waaruit een disk-sector bij de Commodore bestaat. Deze pointers staan aan het begin van iedere sector en worden daar ook neergezet, voordat de rest van de gegevens in het blok wordt geschreven. Daardoor kan de diskdrive-processor en dus de computer bij het teruglezen bepalen op welke sectoren van de diskette de data staan weggeschreven. Bij het uitvoeren van de CLOSE-opdracht wordt bij het schrijven van de laatste sector 254 bytes of minder, ook een EINDADRES als $FF FF naar deze track-sector pointer gezet. Blijft de CLOSE opdracht echter achterwege, dan staan als adres in het pointerwoord de twee bytes van een eventueel nieuwe track-sector opgeslagen, hoewel de data en de $FF FF ontbreken, doordat de CLOSE-opdracht is weggevallen.
Dit is de reden waarom het nogal eens gebeurt, dat twee bestanden op de diskette opeens als één geheel door de diskdrive en de computer worden beschouwd.

De volgende keer zullen we aandacht besteden aan het vervolg van ons Basic-alfabet, waarin opdrachten aan de orde komen als CLR, CMD en CONT.

Jan Bodzinga

Tips & Trucs 64

In deze aflevering komen onder andere de volgende onderwerpen aan bod: reset beveiliging, een data statements genererend programma, de wait instructie, vrije geheugengebieden en natuurlijk weer peeks & pokes.

Reset beveiliging

Om te voorkomen dat een programma illegaal gekopieerd wordt, moet het programma op een aantal punten beveiligd worden. Een daarvan is de beveiliging van de resetknop en de STOP/RESTORE toetsen. Normaal kan een programma onderbroken worden door op STOP/RESTORE of op de resetknop te drukken, waardoor de gebruiker de controle over de computer krijgt en het programma met het SAVE commando kan kopieren. Dit kunnen we tegengaan, door er bijvoorbeeld voor zorgen dat het programma, na het resetten van de computer, niet gestopt wordt, maar gewoon opnieuw begint. Om dit te bewerkstelligen, gebruiken we een slimme truc.

Zodra u op de resetknop drukt, kijkt de computer of er een cartridge in de cartridge poort is geplugd. Als dit het geval is dan start de computer het programma in de cartridge. Is er geen cartridge aanwezig, dan wordt de computer gewoon gereset. Bij het indrukken van de RESTORE toets gebeurt er net zoiets. Eerst wordt er gekeken of er een cartridge aanwezig is. Als dat zo is, dan wordt deze gestart. Anders test de computer of de STOP toets ook is ingedrukt. Als deze is ingedrukt, wordt het programma onderbroken en verschijnt er READY op het scherm. Als de STOP toets niet is ingedrukt, gebeurt er niets. De truc waarmee we de resetknop en de RESTORE toets gaan beveiligen, komt op het volgende neer. We maken de computer wijs dat er een cartridge in de cartridge poort zit en dat het programma, dat we willen beveiligen, in die cartridge is. De computer zal dan na het indrukken van de resetknop of de RESTORE toets dit programma starten.

Als er in de cartridge poort een cartridge zit, dan staat er in de adressen \$8004-\$8008 een identificatiecode waaraan de computer kan zien dat er een cartridge aanwezig is. Deze identificatie code is: CBM80. (let er op dat CBM in hoofdletters is geschreven). Door deze code zelf, in de vorm van ascii waarden, in de adressen \$8004-\$8008 te zetten, denkt de computer dat er een cartridge aanwezig is. De ascii waarden voor de code zijn: \$C3,\$C2,\$CD,\$38 en \$30. We moeten nu alleen nog aan de computer vertellen naar welk geheugenadres hij moet springen, als de resetknop of RESTORE toets wordt ingedrukt. Het adres, waarheen gesprongen moet worden bij het indrukken van de resetknop, moet in de adressen \$8000-\$8001 worden gezet. Het lo-byte van het adres moet in \$8000 worden gezet en het hi-byte in \$8001. Het adres, waarheen gesprongen moet worden bij het indrukken van de RESTORE toets, moet in de adressen \$8002-\$8003 worden gezet. Het adres hoeft natuurlijk niet het beginadres van het te beveiligen programma te zijn. U kunt bijvoorbeeld een speciale machinetaal routine schrijven, die wordt uitgevoerd zodra de resetknop wordt ingedrukt. In dat geval moet u het beginadres van die routine aan de computer doorgeven. Hieronder volgt een klein voorbeeldprogramma.

LABELS MNEMONICS COMMENTAAR

```
ldx #$00  ;Zet
   identificatie-
l1 lda code,x   ;code
   in geheugen
   sta $8004,x
       inx
       cpx #$05
       bne l1
       lda #art ;Zet lo-byte in
       sta $8000    ;adressen
   voor reset
       sta $8002    ;en
   RESTORE
       lda #start ;Zet
   hi-byte in
       sta $8001    ;adressen
   voor reset
       sta $8003    ;en
   RESTORE
       rts
start  inc $d020     ;Verhoog
   borderkleur
       jmp start
code   .byt
   $c3,$c2,$cd,$38,$30
```

Eerst moet u de resetbeveiliging activeren, door het bovenstaande programma een keer te starten. Zodra u nu op de resetknop of RESTORE toets drukt, wordt de routine met de naam 'start' uitgevoerd. De border zal dus continu van kleur veranderen. De enige manier om dit te stoppen is door de computer uit te zetten. Zorg er dus voor dat u de routine op tape of disk gesaved heeft voordat u hem uit probeert.

Data maker

Een aantal afleveringen geleden hebben we de werking van de toetsenbordbuffer uitgelegd. Het is mogelijk om vanuit een Basic programma, opdrachten in de toetsenbordbuffer te zetten, die pas na het beeindigen van het programma worden uitgevoerd. In deze aflevering zullen we een applicatie programma bespreken, dat gebruik maakt van deze techniek. Het programma is een zogenaamde data maker.

Veel Basic programma's maken gebruik van hulproutines, die in machinetaal zijn geschreven. De code van deze routines staat meestal aan het eind van het Basic programma, in de vorm van data statements. Deze data worden dan in het begin van het programma, met behulp van een FOR-NEXT lus en de READ instructie, ingelezen en in het geheugen gepoked. Bij kleine machinetaal routines kan men

Scan-waarden toetsenbord			
0	inst/del	33	i
1	return	34	j
2	crsr hor	35	O
3	f7	36	m
4	f1	37	k
5	f3	38	o
6	f5	39	n
7	crsr ver	40	+
8	3	41	p
9	w	42	l
10	a	43	–
11	4	44	.
12	z	45	:
13	s	46	@
14	e	47	,
15		48	£
16	5	49	*
17	r	50	;
18	d	51	clr/home
19	6	52	
20	c	53	=
21	f	54	ver pijl
22	t	55	/
23	x	56	1
24	7	57	hor pijl
25	y	58	
26	g	59	2
27	8	60	spatie
28	b	61	
29	h	62	q
30	u	63	run/stop
31	v	64	geen toets
32	9		

figuur 1

het geheugen nog wel met de PEEK instructie uitlezen en de zo verkregen waarden met de hand in data statements zetten. Maar bij langere routines moet men toch een vreemde hersenkronkel hebben om dit foutloos te doen en langer dan een kwartier vol te houden. Daarom hebben we onder het motto, "waarom vervelend werk doen, als de computer het ook kan", een klein Basic programma ontwikkeld dat dit vervelende werk van u overneemt. Het programma maakt, zoals al eerder gezegd, gebruik van de toetsenbordbuffer.
Als eerste vraagt de computer om het start- en eindadres van de routine, die u in data statements wilt zetten. Daarna zet de computer alles in data statements. Vervolgens wordt een stukje programma gemaakt dat de data statements met een FOR-NEXT lus in het geheugen poked. Tenslotte wist het programma zichzelf uit, zodat alleen de data statements en het gedeelte dat deze data inleest, overblijven. Hier volgt de listing van de data maker. Zorg er voor dat u het programma saved, voordat u het gaat uitproberen, omdat u het programma anders kwijt bent, nadat het zichzelf gewist heeft.

```
10 rem data maker
100 r=10000
110 input "startadres";sa
120 input "eindadres";ea
130 i=sa
140 if iea then 240
150 print
    "{SHIFT-CLR}{3xCRSR-DOWN}";
    r;"data ";
160 for j=0 to 7
170 if i+jea then 200
180
    a$=str$(peek(i+j)):a$=right
    $(a$,len(a$)-1)
190 print a$",";:next
200 print chr$(20)
210 print
    "r=";r+10;":i=";i+8;":sa=";
    sa;":ea=";ea;":goto 140"
220 print "{HOME}":poke
    198,2:poke 631,13:poke
    632,13
230 end
240 print
    "{SHIFT-CLR}{3xCRSR-DOWN}
    10 for i=";sa;" to";ea
250 print "20 read a:poke
    i,a:next"
260 print "goto 290"
270 print "{HOME}":poke
    198,3:poke 631,13
280 poke 632,13:poke
    633,13:end
290 r=100
300 print
    "{SHIFT-CLR}{3xCRSR-DOWN}";
    :for i=0 to 9
310 print i*10+r:next
320 if r300 then print
    "r=";r+100;":goto 300"
330 poke 198,11
340 for i=631 to 642
350 poke i,13:next
360 print "{HOME}":end
```

Het regelnummer waarop het eerste data statement terecht komt is 10000. Dit nummer kan veranderd worden door de waarde in regel 100 aan te passen. Per regel worden steeds acht datawaarden geplaatst; op de laatste eventueel minder. De stapgrootte tussen twee data regels is 10. Deze grootte kan veranderd worden door het getal 10 in regel 210 te veranderen. Het gedeelte dat de data statements inleest en in het geheugen poked wordt op regel 10 en 20 gezet. Dit kan veranderd worden door regel 240 en 250 aan te passen.
We zullen nu in het kort uitleggen hoe het programma werkt. Variabele R houdt het huidige regelnummer bij. De variabelen SA en EA bevatten respectievelijk het begin- en eindadres van de routine die verwerkt moet worden. De variabele I houdt bij, bij welk adres we zijn met de verwerking.
In regel 140 wordt getest of de hele routine al verwerkt is; als dit zo is, wordt naar regel 240 gesprongen. Regel 150 print een regelnummer met daarachter het woord 'data' op de vierde regel van het scherm. Regels 170-200 printen daarachter acht datawaarden. Regel 210 zorgt ervoor dat alle variabelen de juiste waarden hebben als het programma straks opnieuw gerund wordt. In regel 220 worden twee RETURNS in de toetsenbordbuffer gepoked en in regel 230 wordt het programma beeindigd, zodat de twee regels die nu op het scherm staan, worden uitgevoerd. Regels 240-280 maakt het gedeelte dat de data statements inleest en in het geheugen poked en regels 300-360 wissen het programma.

Vrije geheugengebieden

Het Commodore geheugen is verdeeld in stukken van elk 256 bytes. Een zo'n stuk

geheugen van 256 bytes noemen we een page. Bij elkaar bestaat het geheugen uit 256 pages. Dit klopt ook, want 256 * 256 = 65536. En 65536 bytes vormen 64 K geheugen. We gaan het in dit stukje hebben over de eerste vier pages van de Commodore. In dit gebied, dat van adres 0 tot adres 1023 loopt, bevinden zich namelijk een aantal lege gebieden, waar de programmeur handig gebruik van kan maken.
De eerste page wordt ook wel de zero page genoemd. Aan het eind van deze page bevinden zich vier adressen die niet gebruikt worden door het operating system van de computer. Dit zijn de adressen 251 tot 254 ($FB - $FE). Dit is natuurlijk wel een erg klein geheugengebiedje, maar juist de machinetaalprogrammeur kan hier goed gebruik van maken. Deze geheugenadressen worden in machinetaal meestal gebruikt met zero page adressing, zoals (indirect,x) en (indirect),y adressing.
De tweede page bevat geen lege geheugengebieden. Deze page bevat voor het grootste deel de processor stack. We raden u niet aan om in dit geheugendeel te gaan rotzooien.
Aan het eind van de derde page bevindt zich een redelijk groot geheugengebied dat nergens voor gebruikt wordt. Dit geheugengebied loopt van adres 679 tot adres 767 ($02A7 - $02FF). U kunt in dit gebied bijvoorbeeld een klein machinetaal programma zetten. Maar wat nog veel handiger is, u kunt er een hele sprite in kwijt. U kunt het eerste byte van de sprite op adres 704 zetten en de rest van de sprite op de opeenvolgende adressen. Adres 704 is een veelvoud van 64, namelijk 11 * 64.
Ook de vierde page bevat een klein stukje vrij geheugen en wel van adres 1020 tot 2023 ($03FC - $03FF). Voor deze adressen zit echter de cassettebuffer en deze kan ook prima gebruikt worden om er allerlei rotzooi in te zetten. De cassettebuffer loopt van adres 828 tot adres 1019. U kunt de cassettebuffer natuurlijk alleen als vrij gebied gebruiken als de cassetterecorder niet gebruikt wordt tijdens het gebruik van dit gebied. Bij elkaar heeft u dan een aardig groot gebied vrij. In dit gebied kunnen maar liefst drie sprites worden geplaatst. Deze sprites kunnen worden gezet op de adressen 832 (13 * 64), 896 (14 * 64) en 960 (15 * 64).
Dit waren de lege plaatsen in de laagste vier pages van de computer. Een andere keer zullen we nog lege geheugengebieden bespreken in de rest van het computergeheugen.



Adres 197

In adres 197 wordt bijgehouden wat de laatste toets is, die is ingedrukt. Als er geen toets is ingedrukt, is de inhoud van adres 197 het getal 64. Als er wel een toets is ingedrukt, wordt dat bijgehouden in de laagste vijf bits. In figuur 1 staat een tabel waarin alle waarden van de toetsen staan afgedrukt. U kunt zien dat alle tekens die te maken zijn in combinatie met de shift-, Commodore- of control-toets, niet in de tabel staan. In adres 197 worden deze combinaties allemaal niet bijgehouden. Als u bijvoorbeeld wilt testen of return is ingedrukt, kunt u het volgende statement gebruiken.

```
IF PEEK(197)=1 THEN ...
```

In figuur 1 kunt u zien dat het scan-getal van return 1 is. Er zijn ook een paar lege plaatsen in de tabel. De waarden in de tabel met daarachter een lege plek corresponderen niet met een toets, en kunt u dus ook nooit verkrijgen bij het uitpeeken van adres 197.

De WAIT instruktie

De WAIT instruktie is waarschijnlijk een van de minst gebruikte Basic instrukties. Reden genoeg om er eens wat extra aandacht aan te besteden, want er zijn handige dingen mee te doen. Het formaat van WAIT is de volgende:

```
   WAIT adres,getal1,
getal2
```

U ziet dat WAIT drie parameters heeft. De eerste parameter is een geheugenadres. De tweede en derde parameter moeten allebei getallen zijn. Deze getallen moeten tussen de waarden 0 en 255 liggen. De computer behandelt de WAIT instruktie als volgt. Er wordt eerst naar het opgegeven geheugenadres gekeken. Dat wil zeggen, de waarde die in dat geheugenadres staat wordt opgehaald. Op deze waarde wordt dan eerst een Exclusive OR uitgevoerd met de derde parameter (getal2). Het resultaat hiervan wordt daarna nog vergeleken met de tweede parameter (getal1) door middel van AND. Als het uiteindelijke resultaat 0 (FALSE) is, dan wacht de computer. Als het resultaat een positief getal is, dus ongelijk nul (TRUE), dan gaat de computer verder met de volgende instruktie. Bij uitkomst 0 blijft de computer steeds de bovenstaande berekening uitvoeren, totdat de uitkomst niet nul is. Op deze manier kan er getest worden of een geheugenadres een bepaalde bitcombinatie heeft. Is dit niet het geval, dan wordt er gewacht totdat het geheugenadres wel die bepaalde bitcombinatie heeft.
U zult vermoedelijk wel weten wat een AND van twee getallen inhoudt. Twee getallen worden bit voor bit met elkaar vergeleken. De uikomst van een bitvergelijking is alleen een 1, als beide bits een 1 bevatten. Anders is de uitkomst een 0. Een Exclusive OR tussen twee getallen is geen gewone OR, en zullen we derhalve wat uitgebreider behandelen. Het volgende schema geeft de waarheidstabel van Exclusive OR.

```
bits  uitkomst
--------------
0 0 -   0
0 1 -   1
1 0 -   1
1 1 -   0

Waarheidstabel
    Exclusive OR
```

U ziet dat een bitvergelijking bij een Exclusive OR alleen een 1 oplevert, als slechts een van beide bits op 1 staat. Staan beide bits op 1, dan levert de vergelijking 0 op.
Nu weer terug naar de WAIT instruktie. Voordat we een voorbeeld geven, vertellen we eerst nog even het volgende. De

derde parameter mag weg worden gelaten. De computer zal dan de derde parameter als nul beschouwen. Een Exclusive OR met nul, zal geen invloed hebben. Immers, als een getal wordt vergeleken door middel van Exclusive OR met nul, zal het resultaat weer hetzelfde getal zijn. U kunt dit controleren in de waarheidstabel. Bekijk nu het volgende voorbeeld eens.

```
WAIT 653,2
```

Met geheugenadres 653 kan getest worden of de shift-, Commodore- of controltoets wordt ingedrukt (zie Tips & Trucs April/Mei 1989). De computer kijkt naar de inhoud van adres 653. Er is geen derde parameter meegegeven, dus de computer beschouwt de derde parameter als nul. Hierdoor verandert er niets aan de waarde die de computer heeft opgehaald uit adres 653. Daarna wordt deze waarde geAND met de tweede parameter. Deze parameter heeft de waarde 2, dus er wordt een AND uitgevoerd met 2. Kortom, er wordt naar het tweede bit van adres 653 gekeken. Als het tweede bit aan staat, is de Commodore-toets ingedrukt, anders niet. Als dit bit aanstaat, levert de AND met 2 als uitkomst ook 2 op, en zal de computer niet wachten. Als het bit uitstaat, levert de AND als uikomst 0 op, en zal de computer wachten tot het bit wel aanstaat. WAIT 653,2 heeft dus tot gevolg dat de computer wacht tot de Commodore-toets wordt ingedrukt. We zullen als laatste nog een voorbeeld geven.

```
WAIT 197,63
```

Deze instruktie zorgt ervoor dat de computer wacht totdat er een toets wordt ingedrukt. We hebben gezien (zie andere tip) dat adres 197 een waarde kleiner dan 64 bevat als er een toets ingedrukt is. De derde parameter is wederom niet aanwezig, dus daar hoeven we geen rekening mee te houden. Door de inhoud van adres 197 met 63 te ANDen, wordt er gekeken of er een of meer van de laagste 5 bits aanstaan (dan is er een toets ingedrukt). Is dit het geval, dan levert de AND een positief getal en wordt er niet gewacht. Staan de laagste vijf bits allen uit, dan is er geen toets ingedrukt en moet er gewacht worden. Dit gebeurt dan ook, omdat de AND vergelijking dan nul oplevert.
U ziet dat de derde parameter in de praktijk nauwelijks tot niet gebruikt wordt. Heeft u nog handige WAITs, dan kunt u ze gewoon opsturen.

Peeks & Pokes

186

In dit geheugenadres wordt bijgehouden welk randapparaat het laatst gebruikt is. Een 1 is de cassetterecorder, een 4 is de printer en 8 betekent de disk-drive. Dit zijn enkele voorbeelden van randapparaten, maar er kunnen natuurlijk ook andere apparaten zijn met andere nummers, die in dit adres geregistreerd worden. Als er nog geen randapparaat is gebruik, staat er een 0 in dit adres.



646

Geheugenadres 646 bevat de huidige kleurcode van de cursor. Dit adres kan niet alleen gelezen, maar ook geschreven worden. Om de cursor de kleur geel te geven moet bijvoorbeeld het volgende in worden getypd.

```
POKE 646,7
```

Hierbij hebben de kleuren gewoon de standaard waarden, zoals ze in de user manual staan. Alleen de laagste vier bits van dit adres worden gebruikt voor de kleurcodering. De hoogste vier bits zijn echter wel aangesloten en kunnen dus eventueel gebruikt worden om er iets in op te slaan.

647

Dit adres kan worden uitgepeeked om te kijken wat de kleur is van het karakter onder de cursor. De waarde die zo wordt verkregen hoeft dus niet hetzelfde te zijn als de waarde van de cursorkleur. Van dit adres zijn alle bits aangesloten, zodat een peek een getal tussen de 0 en 255 kan opleveren. Omdat u alleen de laagste vier bits hoeft te hebben, kunt u de volgende regel intypen.

```
PRINT PEEK(647) AND 15
```

157

Adres 157 is een zogenaamde Kernal Message Control Flag. Deze vlag bepaalt wat voor meldingen er worden gegeven bij het gebruik van randapparatuur. Standaard bevat dit adres de waarde 128; de computer is dan in de directe mode. Dit betekent dat er meldingen als SEARCHING, SAVING en FOUND worden gegeven. Dit zijn de Control meldingen.
Door de waarde 192 in dit adres te zetten worden naast de Control meldingen ook Kernal meldingen gegeven. Deze meldingen bestaan uit een foutcode (bijvooorbeeld I/O ERROR 4). Dit kan alleen vanuit machinetaal programma's gebeuren; bijvoorbeeld bij het laden en saven m.b.v. een machinetaal monitor.
De waarde 64 betekent dat er alleen Kernal meldingen worden gegeven. Ook dit is alleen vanuit machinetaal mogelijk.
De waarde 0 onderdrukt alle meldingen. Deze waarde wordt automatisch in adres 157 geplaatst als er een Basic programma draait. De computer is dan in programma mode.

Hylke Sprangers & Michel de Boer

Graphics op de 64

Deel 5: high resolution vervolg

Dit is alweer de vijfde aflevering van de cursus Graphics op de Commodore 64. De wervelende cursus, die de deuren naar de schatkamer van de graphics voor u opent. Reeds bent u langs de duistere wegen van de karakters en de sprites gevoerd. In deze aflevering zullen Michel de Boer en Hylke Sprangers u verder meevoeren in high resolution. Een taai onderwerp, dat wel, maar noodzakelijk als voorbereiding op computerkunst. And now for something completely different...

In de vorige aflevering zijn we begonnen met onze reis door het ruige landschap van hires. Daarin zijn onder meer de volgende onderwerpen aan bod gekomen: opbouw en opslaan van het hires scherm en algoritmen om hires te wissen en om pixels aan te zetten. Omdat hires een vrij ingewikkeld grafisch object is, waar veel over te vertellen is, zullen we in deze aflevering verder gaan met de bespreking van hires. We beginnen ook deze keer met een korte samenvatting van het voorafgaande. Daarna zullen we het gaan hebben over het trekken van lijnen, het maken van cirkels en het inkleuren van vlakken. In de praktijk zult u deze technieken niet zo vaak gebruiken, zeker niet bij het maken van video games. Deze technieken vormen echter een onontbeerlijke basis voor het maken van computerkunst, waar we het in de volgende aflevering over zullen hebben. Ook zullen we het in deze aflevering hebben over multi color hires.

Samenvatting

Voor alle duidelijkheid zullen we in deze paragraaf de basiselementen van hires nog een keer behandelen. Hires is een heel scherm dat geheel uit pixels is opgebouwd. Elk van deze pixels kan afzonderlijk aan en uit worden gezet. Het hires scherm bestaat uit 64000 pixels, 320 pixels horizontaal en 200 pixels vertikaal. Er gaan acht pixels, die horizontaal naast elkaar liggen, in een byte. Zo bestaat het hires scherm uit 64000 / 8 = 8000 bytes. Deze 8000 bytes zijn geordend in groepjes van acht bytes, cellen genaamd. Een cel van het hires scherm komt overeen met een karakter van het lores (tekst) scherm. Het hires scherm bestaat zo uit 1000 cellen. Deze cellen worden op dezelfde manier genummerd als de karakters op het lores scherm. De cel linksboven is dus cel 0, de cel daarnaast cel 1, enzovoort. De eerste acht bytes (byte 0-7) van het hires scherm bevinden zich in cel 0, de tweede acht bytes (byte 8-15) in cel 1. Op deze manier heeft elk byte een eigen bytenummer.
Deze 8000 bytes kunnen op een paar plaatsen in het geheugen worden gezet. Voorlopig hebben we voor het geheugengebied, dat begint op 8192, gekozen. Het geheugenadres van een bepaald byte op het hires scherm kan nu berekend worden door het bytenummer van dat byte bij 8192 op te tellen.
Voor elke cel kan een aparte achter- en voorgrondkleur worden gekozen. Deze twee kleuren worden samengevoegd tot een waarde, die specifiek is voor die achter- en voorgrondkleur. Deze waarde kan als volgt berekend worden: kleurwaarde = 16 * voorgrond kleur + achtergrondkleur. Er zijn dus 1000 kleurwaarden nodig voor de 1000 cellen. Deze 1000 waarden kunnen worden opgeslagen in het geheugengebied dat loopt van adres 1024 tot 2023. Tenslotte kan het hires worden aangezet met de volgende twee pokes:

```
POKE 53265,PEEK(53265) OR 32
    en POKE 53272,PEEK(53272)
    OR 8.
```

Als laatste hebben we het in de vorige aflevering gehad over pixels aanzetten. Dit kan gedaan worden met de volgende drie Basic regels.

```
gby=8192+320*int(yc/8)+8*int
    3(xc/8)+(yc and 7)
jbi=2^(7-(xc and 7))
poke gby,peek(gby) or jbi
```

Adressen voor high resolution	
Adres	**Funktie**
53265	bit 5: 0 = hires uit 1 = hires aan
53272	bit 3: hires pointer 0 = low segment 1 = high segment
53270	bit 4: 0 = single color 1 = multi color
1024-2023	single color: bit 0-3: achtergrondkleur bit 4-7: voorgrondkleur multi color: bit 0-3: kleur met bitcombinatie 10 bit 4-7: kleur met bitcombinatie 01
55296-56295	multi color: kleur met bitcombinatie 11

figuur 1

Hierbij staat xc voor x-coordinaat (0-319) en yc voor y-coordinaat (0-199). Het pixel linksboven heeft coordinaten (0,0) en het pixel rechtsonder (319,199).

Lijnen trekken

We zijn nu aangekomen bij het onderwerp lijnen trekken. Tussen twee willekeurig uitgekozen punten valt, volgens Postulaat 1 van de heer Euclides (+/- 300 v. Christus), een lijn te trekken. Het is nu aan ons om een algoritme te verzinnen om de Commodore een lijn te laten trekken tussen twee punten. Helaas zal het volgende stukje ook weer een wiskundige inslag hebben. We kunnen u toch geruststellen: ook al begrijpt u van alle wiskunde bijna niets of helemaal niets, het gaat er om dat u de voorbeeld programma's goed leert hanteren. De formules die in deze programma's worden gebruikt, willen we wel graag uitleggen, maar het hindert niet als u het allemaal niet volledig kunt volgen. Wel, we gaan het trekken van lijnen uitleggen aan de hand van een illustratie (zie figuur 2). In dit figuur ziet u twee punten (pixels) A en B waartussen een lijn loopt. Het gaat erom dat deze lijn AB door de computer zelf wordt getrokken, als de twee punten A en B worden aangeboden aan de computer. Met andere woorden, de computer krijgt een begin- en eindpunt (hier punt A en B) en moet tussen die twee punten een lijn trekken. Zoals u in de figuur kunt zien, is er een rechthoekige driehoek geconstrueerd met lijnstuk AB als hypotenusa. De hypotenusa is de zijde die tegenover de rechte hoek ligt.

We voeren nu het begrip richtingscoefficient (rico) in. De rico van een lijn is de vertikale afstand tussen twee willekeurige punten op die lijn gedeeld door de horizontale afstand van die twee zelfde punten. We willen de rico hebben van lijnstuk AB. Daartoe delen we de vertikale afstand tussen A en B door de horizontale afstand tussen A en B. In het plaatje: rico AB = yl / xl. In ons plaatje is de rico van lijnstuk AB ongeveer 2. Oftewel, yl is bijna twee keer zo groot als xl. Dit betekent dat bij het tekenen van lijnstuk AB bij elke stap naar rechts, twee stappen naar boven gedaan moeten worden. U ziet dat een lijn volledig wordt bepaald door een punt waar de lijn doorheen gaat en door zijn rico. Als we twee punten hebben waar een lijn doorheen moet, moet de rico van de lijn worden uitgerekend. Vervolgens kan betrekkelijk eenvoudig de lijn worden getrokken. Het komt erop neer dat we steeds 1 stap in horizontale richting doen en rico (=yl/xl) stappen in vertikale richting. Zo komen we elke keer bij het volgende pixel op de lijn. Als de rico groter dan 1 is maken we steeds 1 stap in vertikale richting en xl/yl stappen in horizontale richting; anders ontstaat er een stippellijn. Op deze manier zorgen we ervoor dat de stapgrootte in zowel de x- als y-richting altijd kleiner of gelijk 1 is, waardoor elk volgend pixel aansluit bij het vorige pixel. We zullen het programma om een lijn te trekken alvast geven.

```
5 rem lijn trekken
10 bx=10:by=20:ex=230:ey=120
20 gl=abs(bx-ex)
30 if abs(by-ey)>gl then
   gl=abs(by-ey)
40 xs=(bx-ex)/gl:ys=(by-ey)/
   gl
50 for x=1 to gl
60 xc=int(bx):yc=int(by)
   :gosub 100
70 bx=bx+xs:by=by+ys:next
80 goto 80
100  gby=8192+int(yc/8)*320+
   int(xc/8)*8+(yc and 7)
110 gbi=2^(7-(xc and 7))
120 poke gby,peek(gby) or
   gbi:return
```

In dit programma wordt het hires scherm niet aangezet en gewist, dat moet u zelf doen. We zullen het programma zo goed mogelijk proberen uit te leggen. Omdat een lijn is opgebouwd is uit aparte pixels, moeten we een subroutine hebben die een willekeurige pixel aanzet. Die subroutine bevindt zich op de regels 100 tot 120. In regel 10 worden de twee punten gedefinieerd, waartussen de lijn wordt getrokken. De punten hebben de coordinaten (bx, by) en (ex, ey). In de regels 20 en 30 wordt de afstand bepaald tussen de x- (horizontale afstand) en de y-coordinaat (vertikale afstand) van de twee punten. De grootste afstand wordt aan de variabele gl toegekend. Er wordt verder met de grootste afstand (gl) gerekend om te voorkomen dat er een stippellijn ontstaat. Daarna wordt in regel 40 de stapgrootte bepaald in de x- (xs) en y-richting (ys). De lijn wordt nu als volgt getekend. We beginnen vanuit het eerste opgegeven punt met de coordinaten (bx,by). We tekenen het punt. Het tekenen van een pixel gebeurt in de subroutine vanaf regel 100. Daarna worden de coordinaten van het punt verhoogd met xs in de x- en ys in de y-richting. Dan wordt het volgende pixel aangezet. Dit alles gebeurt in de lus in regels 50 en verder. U kunt in het programma zien dat altijd of xs of ys de waarde 1 of -1 heeft. De x- of y-coordinaat wordt dan verhoogd of verlaagd met 1. Op deze manier kunnen we redelijk nauwkeurig een lijn tekenen. Het programma bevat een kleine onnauwkeurigheid. Als we het beginpunt van de lijn gelijkmaken aan het eindpunt van de lijn, wordt er voor gl de waarde 0 gevonden. Vervolgens wordt er dan bij het berekenen van xs en ys gedeeld door 0. Dit kan niet, en de computer zal stoppen met een error. Hoewel dit randgeval makkelijk te verhelpen is met een IF statement, hebben we daar toch vanaf gezien, omdat een nul dimensionale lijn niet gedefinieerd is.

Het testen van pixels

Als je een willekeurige pixel hebt, is het soms handig om te kijken of dat pixel aan of uit staat. Dat dit werkelijk handig is zullen we straks nog zien bij het inkleuren van vlakken. In feite is dit testen van een pixel heel eenvoudig. We gaan weer uit van een pixel met coordinaten (XC, YC). Het eerste wat we dan moeten doen, is het berekenen van GBY en JBI. Daarbij staat GBY voor het goede byte, waarin het pixel zich bevindt. JBI is het juiste bit in dit byte. De formules voor deze twee variabelen zullen we hier niet meer geven (deze staan onder andere in de samenvatting aan het begin van dit artikel).
Om nu te testen of een pixel uit staat, kan het volgende statement gebruikt worden.

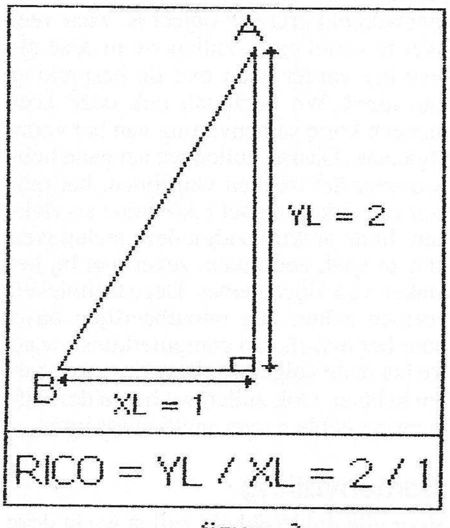
```
IF (PEEK(GBY) AND JBI)=0 THEN
   REM PIXEL UIT
```

Het volgende statement kan gebruikt worden om te testen of een pixel aan staat.

```
IF (PEEK(GBY) AND JBI)<>0
   THEN REM PIXEL AAN
```

Het inkleuren van vlakken

Het applikatie programma van vorige keer omvatte een reeks hires routines. Bij deze routines zat ook een FILL funktie. Hiermee konden omsloten vlakken ingekleurd worden. Het enige wat gedaan moest worden, was het meegeven van een willekeurig punt in het vlak dat ingekleurd moest worden. We zullen in deze paragraaf uitleggen hoe een routine gebouwd kan worden, die een vlak inkleurt.



figuur 2

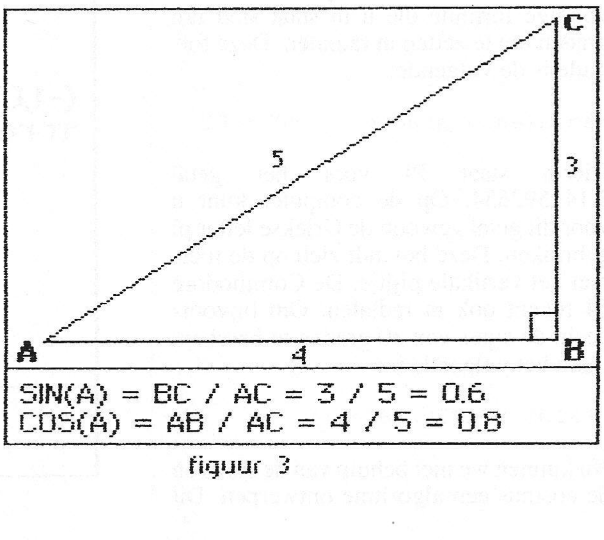
We zullen gelijk maar een Basic programma geven, voor het inkleuren van vlakken. Deze versie werkt ongeveer op dezelfde manier als de machinetaalroutine in de vorige aflevering. Hier komt het programma.

```
10 rem inkleuren vlakken
20 xc=xc-1:gosub 170
30 if (peek(gby) and jbi)=0
   then 20
40 xc=xc+1:gosub 190
50 yc=yc-1:gosub 170
60 if (peek(gby) and jbi)=0
   and (pd=0) then
   pd=1:gosub 160
70 if (peek(gby) and jbi)<>0
   and (pd=1) then pd=0
80 yc=yc+2:gosub 170
90 if (peek(gby) and jbi)=0
   and (pu=0) then
   pu=1:gosub 160
100 if (peek(gby) and jbi)<>0
   and (pu=1) then pu=0
110 yc=yc-1:xc=xc+1:gosub
   170:xc=xc-1
120 if (peek(gby) and jbi)=0
   then 40
130 if sp=0 then return
140 pu=0:pd=0:sp=sp-1
150 xc=sx(sp):yc=sy(sp):goto
   20
160 sx(sp)=xc:sy(sp)=yc:sp=
   sp+1:return
170 gby=8192+320*int(yc/8)+
   8*int(xc/8)+(yc and 7)
180 jbi=2^(7-(xc and
   7)):return
190 gosub 160:poke
   gby,peek(gby) or
   jbi:return
```

Voordat dit programma wordt aangeroepen, moet er eerst een waarde worden toegekend aan de variabelen XC en YC. Het aanroepen van dit programma moet gebeuren met GOSUB 10. Het programma is een subroutine, die ingebouwd kan worden in een ander programma. (XC, YC) zijn de coordinaten van een punt in het vlak, dat ingekleurd moet worden. Vanaf dit opgegeven punt wordt er in de regels 20 en 30 in een rechte, horizontale lijn naar links gelopen. Dit gebeurt door XC, de x-coordinaat van het punt, steeds met een te verlagen. Dit gaat door totdat er een scheidslijn is gevonden, oftewel een pixel dat aan staat. In het programma bevinden zich een paar subroutines. De subroutine op regel 170 berekent GBY en JBI, het byte en bit in dat byte, die corresponderen met het pixel (XC, YC). Door deze routine eerst aan te roepen, kan er in regel 30 getest worden of het pixel aan staat. Als dit niet het geval is, wordt XC weer verlaagd en het volgende pixel getest.
Als de scheidslijn links gevonden is, wordt vervolgens in regel 40 het eerste pixel aangezet, door de subroutine op regel 190 aan te roepen. Er wordt een lijn naar rechts getrokken, totdat er een scheidslijn aan de rechterkant is gevonden. Tijdens het trekken van deze lijn, wordt er boven en onder de lijn gekeken of de pixels daar aan of uit staan. Pixels die uit staan, moeten immers straks ook opgevuld worden. Als er een pixel wordt gevonden die uit staat, wordt die bewaard, om later opgevuld te worden. Het bewaren van de coordinaten van zo'n pixel doen we door middel van een stack. Een stack is als het ware een rij van gegevens. Als er een gegeven toe wordt gevoegd aan de rij, wordt dat gegeven achter aan de rij geplaatst. Als er een gegeven uit de rij moet worden gehaald, kan alleen het achterste gegeven worden gepakt. Dit is ook het gegeven wat het laatst aan de rij is toegevoegd. Zo'n rij noemen we een stack. Met gegeven bedoelen we in dit geval de coordinaten van een pixel. Door de subroutine op regel 160 aan te roepen, worden XC en YC op de stack gezet. Voor XC is er de stack met naam SX en voor YC de stack met naam SY. SX en SY zijn in feite gewoon twee arrays. Het laatste element in het array wordt aangewezen door de zogenaamde stackpointer SP. Door SP weten we ook altijd hoeveel elementen er op de stack staan. Als er nog niets of niets meer op de stack staat, is SP gelijk aan nul.
We gaan weer even terug naar regel 40. Daar werd begonnen met het trekken van de lijn naar rechts. In regel 50 wordt de y-coordinaat verlaagd, om in regel 60 onder de lijn te kijken naar pixels die uit staan. Er wordt ook nog een variabele PD bijgehouden. Deze wordt op een gezet, als er een pixel is gevonden dat uit staat. Dit, omdat het pixel daarnaast dan niet meer onthouden hoeft te worden, als deze ook uit staat. Als PD dus op 1 staat, wordt er niet meer onder de lijn gekeken. Als PD echter op 0 staat, wordt dit wel gedaan. Dit alles gebeurt in de regels 60 en 70. In regel 80 wordt de y-coordinaat met twee verhoogt, om nu boven de lijn te kijken. Dit gebeurt in de regels 90 en 100. In plaats van de variabele PD, wordt nu de variabele PU bijgehouden, om precies dezelfde redenen. In regel 110 en 120 wordt weer verder gegaan met het trekken van de lijn naar rechts. Er wordt daar gekeken of er al een rechter scheidslijn is. Zoniet, dan wordt er teruggesprongen naar regel 40, om het volgende pixel aan te zetten.
Als de lijn wel klaar is, wordt er in regel 130 gekeken of er nog een pixel op de stack staat. Als dit niet het geval is, staat SP op 0, en wordt er gestopt, d.w.z. terugspringen naar de plaats vanwaar de routine werd aangeroepen. Het vlak is dan ingekleurd. Anders worden in de regels 140 en 150 de coordinaten van een pixel van de stack gehaald. Het hele verhaal begint dan weer opnieuw. De coordinaten worden weer naar links verplaatst, totdat er een scheidslijn wordt gevonden. Er wordt daarna weer begonnen met het trekken van een lijn naar rechts.
U moet er wel aan denken, dat het vlak dat u wilt inkleuren, geheel afgebakend is door lijnen. Als u gewoon het hires aanzet en bovenstaande routine aanroept, gebeurt er niets. Er moet echt een vlak zijn dat ingekleurd kan worden.

Cirkels

Veel meetkundige figuren, zoals driehoeken en rechthoeken, kunnen gemaakt worden door een aantal rechte lijnen te trekken. Voor dergelijke figuren is het dus voldoende om het algoritme voor het trekken van een lijn te kennen. Dit algoritme was een soort wiskundige lineaal, die ons in staat stelde om lijnen op het scherm te trekken. Om cirkels te tekenen, hebben we echter niets aan een lineaal; we hebben een passer nodig. Daarom hebben we voor het maken van cirkels een nieuw algoritme nodig, dat als het ware een wiskundige passer is.
Het algoritme om cirkels te tekenen is helaas wat ingewikkelder dan het algoritme om een lijn te trekken. Allereerst voeren we twee termen in. De eerste is het middelpunt: dit is het gaatje wat u in het papier prikt als u een cirkel met een passer tekent. De tweede is de straal: dit is de afstand tussen de twee benen van de passer, ofwel de afstand tussen het middelpunt en de omtrek van de cirkel. Voor de wiskundig geinteresseerde lezers zullen we nu proberen om het algoritme zo helder mogelijk uit te leggen. Degenen die niet zo dol zijn op wiskunde en alleen geinteresseerd zijn in het resultaat, de cirkel



figuur 3

op het scherm, kunnen het wiskundige gedeelte overslaan en genoegen nemen met het Basic programma om cirkels te tekenen.

Het algoritme is gebaseerd op de wiskundige functies sinus en cosinus. We zullen daarom eerst een korte uitleg van deze twee functies geven. Beide functies delen twee zijden van een rechthoekige driehoek op elkaar. Een rechthoekige driehoek is een driehoek waarvan een van de hoeken 90 graden is. Die hoek wordt een rechte hoek genoemd. In figuur 3 staat zo'n driehoek. De sinus en de cosinus zijn functies die worden uitgevoerd op de grootte van een hoek van de driehoek, net zoals worteltrekken een functie is, die wordt uitgevoerd op een positief getal. We kunnen bijvoorbeeld de sinus van hoek A berekenen. Dit noteren we als SIN(A). Deze berekening is heel simpel; de sinus van een hoek deelt namelijk de overstaande zijde door de schuine zijde. Met de schuine zijde wordt de zijde, die tegenover de rechte hoek (hoek B) ligt, bedoeld; in ons voorbeeld dus zijde AC. Met de overstaande zijde bedoelen we de zijde die tegenover de hoek, waarvan we de sinus willen berekenen (hoek A), ligt; hier dus zijde BC. De uitkomst van SIN(A) is dus 3 / 5 = 0,6.
Ook kunnen we de cosinus van hoek A berekenen. Dit noteren we als COS(A). Deze berekening is net zo makkelijk als de berekening van de sinus. Nu moet de aanliggende zijde door de schuine zijde gedeeld worden. Met de aanliggende zijde wordt de zijde die tussen de hoek (hoek A) en de rechte hoek (hoek B) ligt, bedoeld. De uitkomst van COS(A) is dus 4 / 5 = 0,8.
Normaal wordt de grootte van een hoek in graden gemeten. Dit kunt u bijvoorbeeld met een geodriehoek of gradenboog doen. U kunt daarop dan aflezen hoe groot een bepaalde hoek is. Helaas worden hoeken in de wiskunde niet in graden gemeten, maar in radialen. Gelukkig is er een eenvoudige formule die u in staat stelt om graden om te zetten in radialen. Deze formule is de volgende:

```
radialen = graden / 180 * PI
```

Hierin staat PI voor het getal 3.141592654. Op de computer kunt u voor dit getal gewoon de Griekse letter pi gebruiken. Deze bevindt zich op de toets met het vertikale pijltje. De Commodore 64 rekent ook in radialen. Om bijvoorbeeld de sinus van 30 graden te bereken, tikt u het volgende in:

```
PRINT SIN(30/180*PI).
```

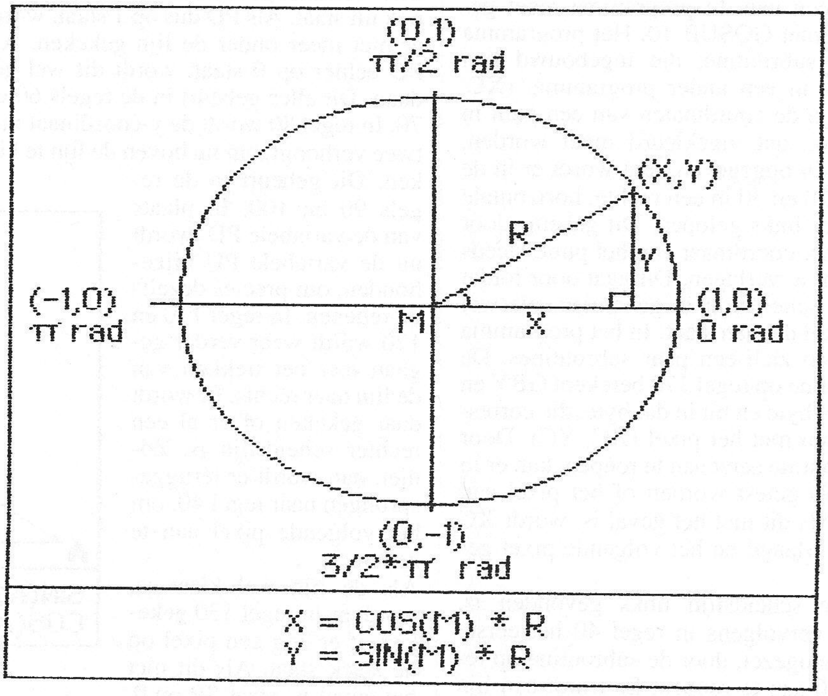
Nu kunnen we met behulp van de sinus en de cosinus een algoritme ontwerpen. Dit doen we aan de hand van figuur 4. Daarin staat een cirkel getekend. Het middelpunt hebben we M genoemd. De lijn R is de straal van de cirkel. De letter 'R' komt van het Latijnse woord 'radius', dat straal betekent. Het middelpunt van de cirkel heeft de coordinaten (0,0). De straal van de cirkel heeft lengte 1. Met deze twee gegevens kunnen we de hele cirkel tekenen. We moeten daarvoor van elk punt op de cirkel de coordinaten berekenen en vervolgens het punt aanzetten. Van vier punten zien we meteen wat de coordinaten zijn (deze staan in de figuur vermeld). Een daarvan is punt (1,0). Vanaf dit punt beginnen we de cirkel tegen de klok in te tekenen. Het berekenen van de coordinaten van de punten gaat als volgt. We kunnen bij elk punt een rechthoekige driehoek construeren, met M als een van de hoeken en R als schuine zijde. In figuur 4 hebben we zo'n driehoek bij een willekeurig punt (X,Y) geconstrueerd. Het is nu makkelijk in te zien dat de aanliggende zijde van hoek M lengte X heeft en dat de overstaande zijde lengte Y heeft. Om de waarde van X en Y te bepalen, moeten we dus de lengte van deze twee zijden berekenen. Dit gaan we doen met de sinus en de cosinus van hoek M. De uitkomsten van deze twee functies zijn de volgende:

```
SIN(M) = Y / R
COS(M) = X / R
```

Hieruit kunnen we vervolgens Y berkenen door SIN(M) met R te vermenigvuldigen, want SIN(M) * R = Y / R * R = Y. Op dezelfde wijze wordt X verkregen door COS(M) met R te vermenigvuldigen. We kunnen nu een hele cirkel tekenen door bij een hoek van 0 radialen te beginnen en de hoek vervolgens steeds een beetje te vergroten (door er bijvoorbeeld elke keer PI/30 bij op te tellen). Van vier punten hebben we in de figuur de hoek vermeld in radialen. U kunt zien dat we de hoek moeten vergroten tot we bij 2*PI radialen (360 graden) zijn, omdat we dan de hele cirkel doorlopen hebben. De punten die we op deze wijze berekenen grenzen niet allemaal aan elkaar. Het is daarom niet voldoende om de berekende punten aan te zetten; er zou dan een stippellijn ontstaan. Dit kan verholpen worden door tussen het berekende punt en het voorgaande punt een lijn te trekken.
In ons voorbeeld hebben we als middelpunt het punt (0,0) gekozen. Op de computer willen we natuurlijk elk willekeurig punt als middelpunt kunnen kiezen. Dit kunnen we bereiken door eerst de coordinaten ten op zichte van punt (0,0) te bereken en deze vervolgens naar het goede punt te verschuiven. Met enig wiskundig inzicht kan ingezien worden dat, voor een cirkel met een willekeurig middelpunt, de formules voor X en Y zijn:

```
X = COS(M) * R + MX
Y = SIN(M) * R + MY
```

Hierbij zijn MX en MY respectievelijk de x- en de y-coordinaat van het middelpunt. Als u met deze twee formules een cirkel tekent, zal blijken dat de cirkel niet perfect rond is, maar een beetje ovaal. Dit komt doordat een pixel niet vierkant, maar rechthoekig is. U kunt wel een perfecte cirkel tekenen door de straal R in de formule voor X met 1.1 te vermenigvuldigen. De formule wordt dan: X = COS(M)



figuur 4

* R * 1.1 + MX. Door in de formules voor X en Y twee verschillende waarden voor R in te vullen, kunt u i.p.v. cirkels ellipsen tekenen. Hieronder volgt de Simon's Basic listing om een cirkel te tekenen. In regel 50 moet u voor {PI} gewoon de Griekse letter gebruiken.

```
5 rem cirkels tekenen
10 input "straal";r
20 input "middelpunt
   (mx,my)";mx,my
30 hires 0,15
40 x1=mx+1.1*r:y1=my
50 for m=0 to 2*{PI} step
   {PI}/30
60 x2=1.1*r*cos(m)+mx:y2=
   r*sin(m)+my
70 line x1,y1,x2,y2,1
80 x1=x2:y1=y2:next
90 goto 90
```

Het programma vraagt de lengte van de straal en daarna de coordinaten van het middelpunt. Vervolgens tekent de computer de cirkel.
In regel 40 staat nog een formule die we niet besproken hebben. Deze formule zorgt ervoor dat X1 en Y1 de coordinaten van het punt rechts van het middelpunt zijn, zodat er bij de berekening van het eerste punt in regel 60 altijd een voorganger is, waarnaar een lijn kan worden getrokken.

Simon's Basic

In de listing voor het tekenen van een cirkel, hebben we twee Simon's Basic instructies gebruikt. We hadden hiervoor ook de hires routines uit de vorige aflevering kunnen gebruiken. Dit hebben we niet gedaan omdat het programma door de SYS instructies moeilijker leesbaar wordt. Voor degenen die geen Simon's Basic hebben, maar wel de hires routines uit de vorige aflevering, zullen we aangeven wat u in de listing moet veranderen.
Regel 30 moet veranderd worden in:

```
30 poke 53265,peek(53265) or
   32: poke
   53272,peek(53272) or 8
31 sys 49264:sys 49225,15,0
```

Regel 70 moet veranderd worden in:

```
70 sys 49186,x1,y1,x2,y2,1
```

Sprites op hires

Net zoals op een tekst scherm kunnen er ook sprites op een hires scherm worden geplaatst. De regels voor overlapping en botsing met pixels zijn precies hetzelfde als bij overlapping en botsing met karakters. We zullen ze daarom niet opnieuw behandelen. Een uitgebreide beschrijving

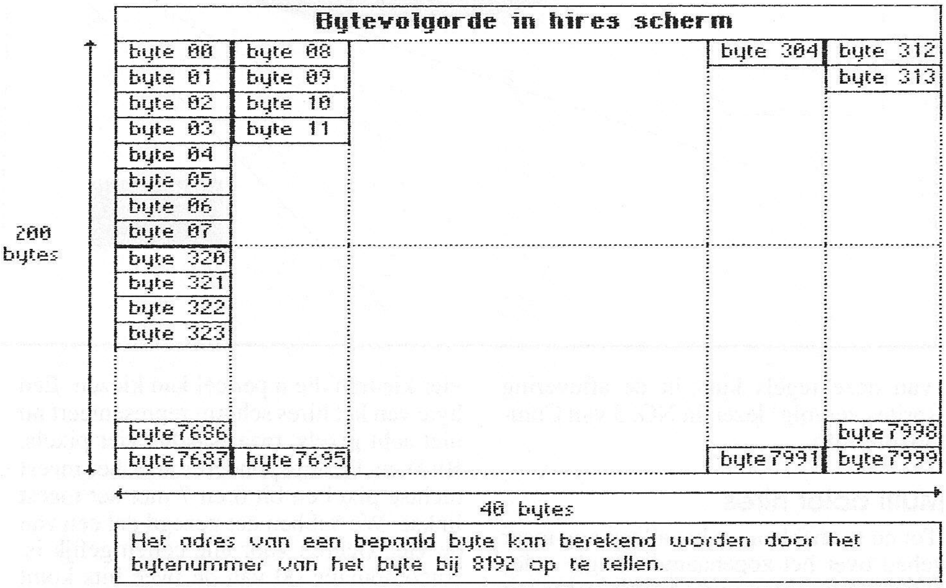
Rectificatie graphics

In het vorige nummer (No. 6, sept/okt 1989) is er per ongeluk wat misgegaan. Bij het artikel "Graphics op de 64" zijn een viertal illustraties niet afgedrukt. Deze illustraties zijn echter noodzakelijk voor een goed begrip van de tekst. We hebben deze illustraties alsnog in deze aflevering geplaatst.

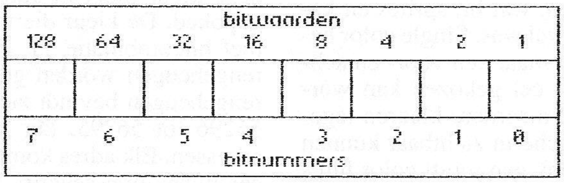
Figuur 1 laat een gedeelte van het hires scherm zien. Het plaatje verduidelijkt de opbouw van het hires scherm.
Figuur 2 toont de ordening van de bits in een byte.

In figuur 3 staat een tabel, waarmee u de codes van de voor- en achtergrondkleuren van een hires scherm kunt bepalen.
Het laatste figuur, figuur 4, bevat een tabel die nodig is voor het gebruik van het applicatie programma, "Hires Routines".

Ook bij het artikel "Tips & Trucs" is er een illustratie weggevallen. Deze illustratie moest de waarheidstabellen van AND, OR en NOT bevatten. Daar deze illustratie niet puur noodzakelijk is voor het volledig begrijpen van de tip, wordt deze niet meer geplaatst.
Excuses voor het geleden ongemak.



figuur 1



figuur 2

Kleur	Voorgrond waarde	Achtergrond waarde
zwart	0	0
wit	16	1
rood	32	2
cyaan	48	3
paars	64	4
groen	80	5
blauw	96	6
geel	112	7
oranje	128	8
bruin	144	9
licht rood	160	10
grijs 1	176	11
grijs 2	192	12
licht groen	208	13
licht blauw	224	14
grijs 3	240	15

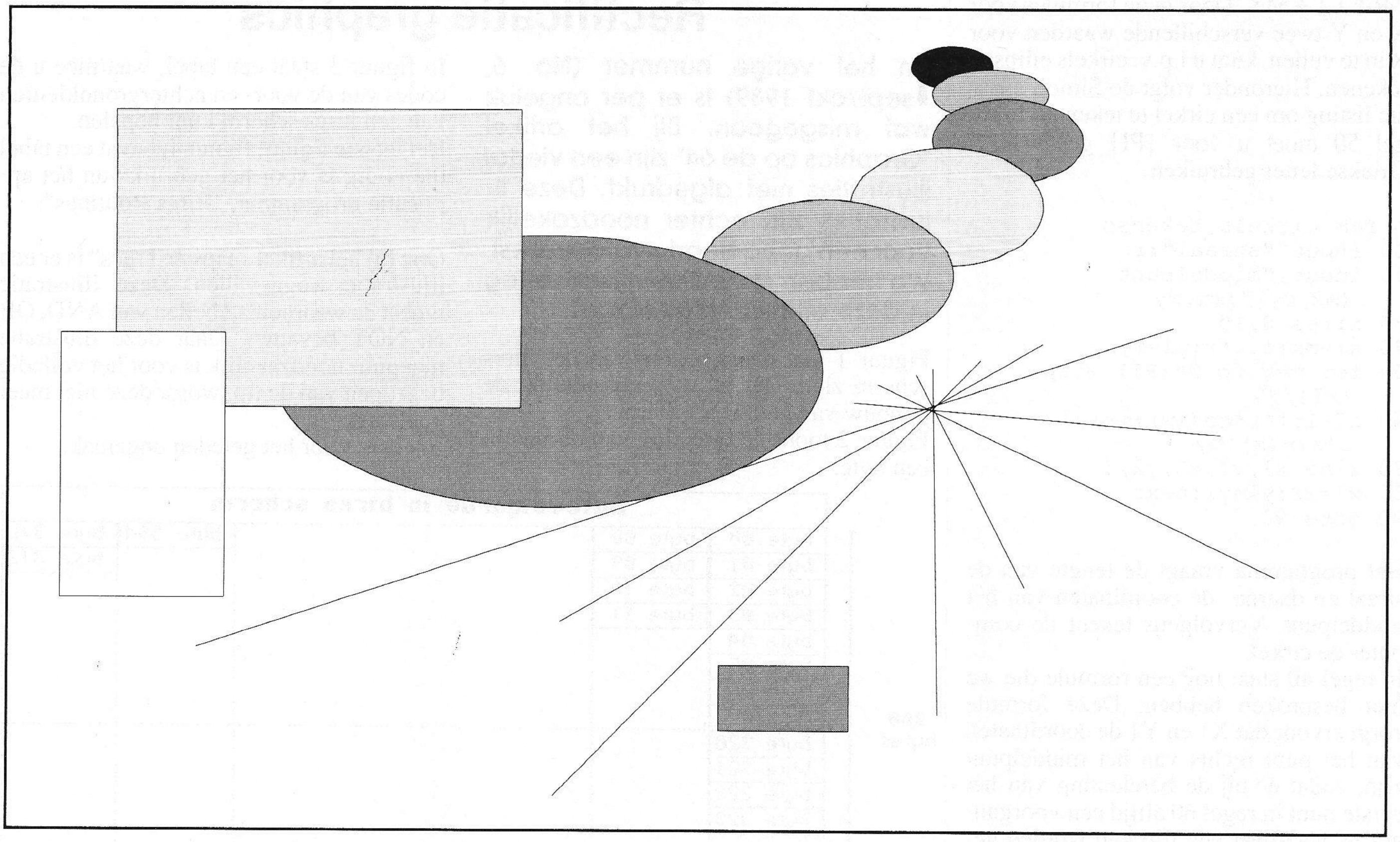
figuur 3

Startadressen van de hiresroutines

routine	machine taal	basic	parameters
DOT	$C085	49162	X,Y,MODE
LINE	$C0AE	49186	X1,Y1,X2,Y2, MODE
CLEAR	$C314	49264	
COLOUR	$C336	49225	Achtergrond, Voorgrond
TEST	$C358	USR	USR(X),Y
FILL	$C386	49240	X,Y,MODE

Vanuit Basic moeten de parameters achter het SYS commando worden meegegeven.
Vanuit machinetaal moeten de parameters, voor de aanroep van de routine, achterelkaar op adres $02A7 en opvolgende adressen worden gezet

figuur 4



van deze regels kunt in de aflevering 'sprites vervolg' lezen in NO. 5 van Commodore Info.

Multi color hires

Tot nu toe hebben we het alleen nog maar gehad over het zogenaamde single color hires. Single color houdt echter niet in dat er maar een kleur op het hires scherm gebruikt kan worden, wat bij sprites en karakters wel het geval was. Single color betekent hier, dat er maar een voor- en achtergrondkleur per cel gekozen kan worden, zodat er wel meerdere kleuren tegelijk op het hires scherm zichtbaar kunnen zijn. Er bestaat ook een multi color hires mode. Deze staat toe, dat er per cel vier verschillende kleuren gekozen kunnen worden. Helaas moet een van deze kleuren voor alle cellen gelijk zijn. De andere drie kleuren kunnen per cel afzonderlijk worden gekozen.

Het gebruik van de multi color mode loopt bij de drie grafische objecten vrijwel parallel. Net zoals bij karakters en sprites, wordt ook deze keer een pixel gerepresenteerd door twee bits. Hierdoor wordt het scherm grover, omdat het scherm nu nog maar 160 pixels in horizontale richting heeft. In de vertikale richting heeft het scherm nog steeds 200 pixels. En ja, u begrijpt het al, met deze twee bits kunnen vier bitcombinaties worden gemaakt, te weten 00, 01, 10 en 11. Deze vier bitcombinaties corresponderen natuurlijk met de vier kleuren die u per cel kan kiezen. Een byte van het hires scherm representeert nu niet acht pixels, maar slechts vier pixels. Bit 0 en 1 corresponderen met het meest rechtse pixel en bit 6 en 7 met het meest linkse. We hebben net gezegd dat een van de vier kleuren voor alle cellen gelijk is. Bitcombinatie 00 van de twee bits komt overeen met deze kleur. De waarde van deze kleur moet in adres 53281 worden gepoked. De kleur die wordt aangegeven met bitcombinatie 11, moet in het kleurengeheugen worden gepoked. Dit kleurengeheugen bevindt zich op de adressen 55296 tot 56295. Dit zijn precies 1000 adressen. Elk adres komt overeen met een cel uit het hires scherm. Voor elke cel kan dus een andere kleur worden gekozen, die in die cel wordt aangegeven met bitcombinatie 11. Nu hebben we nog twee bitcombinaties over, namelijk 01 en 10. Ook voor deze twee bitcombinaties kan per cel een andere kleur worden gekozen. Deze twee kleuren hebben echter niet allebei een apart geheugengebied, maar worden weer samengenomen tot een waarde, net zoals bij single color hires. Deze waarde kan berekend worden met de volgende formule:

```
waarde = 16 * kleur 01 +
    kleur 10
```

Kleur 01 is de kleur die wordt aangegeven met bitcombinatie 01 en kleur 10 is de kleur die wordt aangegeven met bitcombinatie 10. De berekende waarde moet in het schermgeheugen worden gepoked. Dit schermgeheugen bevindt zoch op de adressen 1024 tot 2023. Elk adres correspondeert weer met een cel.

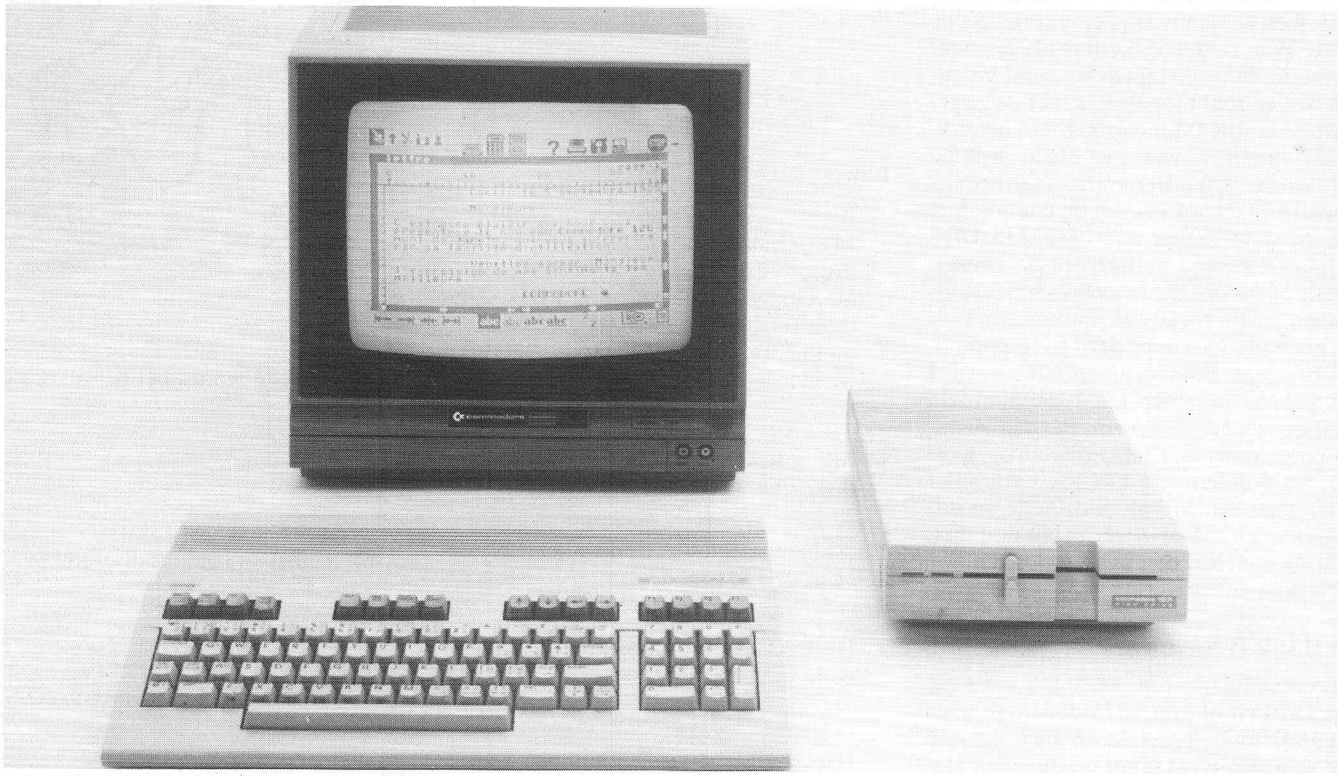
Tenslotte moeten we de computer in de multi color mode brengen. Hiervoor moet bit 4 van adres 53270 worden aangezet. Dit kunt u doen met

```
POKE 53270,PEEK(53270) OR 16.
```

He, he, eindelijk zijn we door de taaie theorie heen. We dachten zo dat u nu wel een redelijke basis had op het gebied van high resolution. Als hulp bij het programmeren hebben we alle adressen die te maken hebben met hires in figuur 1 afgebeeld. Zoals we al hebben aangegeven heeft u deze basis nodig bij het onderwerp 'Computer Art'. Aan dit onderwerp zullen we een aantal afleveringen besteden. De volgende aflevering zal in zijn geheel gewijd zijn aan Computer Kunst. Hoi, hoi!

Michel de Boer & Hylke Sprangers

Diskdrives voor de 128



Als we het over randapparatuur voor computers hebben dan denken we natuurlijk eerst aan een diskdrive. Immers wat heb je nu aan een computer zonder een diskdrive? vrijwel niets.'Maar je kunt toch ook een datarecorder gebruiken', horen we nu een aantal mensen roepen. Jazeker dat kan, je kunt ook het kanaal over zwemmen om in Engeland te komen, maar waarom zou je moelijk doen als het makkelijk kan. Gelukkig zag commodore dit ook in en ontwierp destijds al een diskdrive voor de VIC-20 (weet u nog?).

Sinds de VIC 20 is er voor elke commodore machine een diskdrive ontworpen en op de markt gebracht. Zo ook voor de commodore 128. Er zijn er twee, namelijk de 1570 en de 1571. Het verschil tussen beide is dat de een enkel zijdige diskettes gebruikt en de andere dubbelzijdige.
In dit artikel nemen we een aantal interressante mogelijkheden van deze drive's onder de loep, want de 1570 en de 1571 kunnen meer dan menig 128 gebruiker weet. Hoe zijn nu het ontwerp van deze diskdrives tot stand gekomen ? Daarvoor moeten we helemaal terug naar de 'prehistorie' voor wat computers betreft namelijk terug naar de CBM 4040 drive's. Dat zijn die oude robuust uigevoerde 'bakken' met twee drive's er in. Deze diskdrive beschikte over twee microproccesoren, een voor de databeheer en de ander voor de drive besturing het zogenaamde multi-proccesor systeem. Dit systeem van twee proccesors moest het systeem sneller maken. Toen kwam commodore met de VIC-20 op de markt en daar moest natuurlijk ook een diskdrive bij. En waarom zou je een geheel nieuwe diskdrive laten ontwerpen als je al een hebt die prima werkt, moeten de experts bij commodore gedacht hebben. En dus werd de 4040 drive eens flink onder handen genomen, we slopen er een drive bij weg en bouwen de DOS (het bestuurings systeem dat in een rom in de diskdrive zit opgeslagen) eventjes om en hoppa, de 1540 diskdrive was geboren.
De 1540 heeft één proccesor, de andere die wel in de 4040 aanwezig was werd dus gewoon weggelaten. Het gevolg hiervan was dat die ene micro-proccesor alle taken moest uitvoeren die voorheen door twee proccesors werden uitgevoerd, en u begrijpt al dat dat de snelheid niet tengoede komt. Hierna kwam de C-64 op de markt en hierbij moest natuurlijk ook weer een diskdrive komen. Vervolgens werd, onder het motto 'waarom zouden we moelijk doen als het makelijk kan', de 1540 open geschroeft en eventjes omgebouwd in een 1541.
De DOS van de 1541 is praktisch gelijk aan die van de 1540 alleen werden de routines voor de seriële bus aangepast omdat de de klokfrequentie van deze C-64 ietsjes lager ligt als die van de VIC-20. Hierdoor werkt de besturing van de seriële bus nog weer langzamer en hiermee het werken met de 1541 natuurlijk. De DOS voor de 1570 en de 1571 diskdrive bestaat uit 16 Kbyte. 8 Kbyte van de 1541 plus 12 Kbyte nieuw besturing systeem. En u raad het alweer, 'we passen dit systeempje eventjes aan en dit proppen we er ook bij en zie hier de DOS voor de 1570 en de 1571 is klaar' moeten ze bij commodore gedacht hebben. De nieuwste versie de DOS v3.0 maakt de chaos compleet. Dit

besturingssysteem is bedoeld voor twee drives met een multiproccesorsysteem. Kortgezegt komt het er op neer dat de DOS van de 1570 en de 1571 een zootje bij elkaar geschraapte routines zijn.
Nu moet u niet meteen uw 1570 of 1571 in de afvalbak gooien want zo erg is het nu allemaal ook weer niet. Als je maar weet waar de foutjes zitten kun je die zo ontwijken en dan is er geen vuiltje aan de lucht. Wat is de bedoeling nu precies? Welnu in dit artikel gaan we eerst uitleggen wat er zoal mogelijk is met de zogenaamde USER 0 functie's. En in het volgende nummer van uw eigen lijfblad plaatsen we een programma waarmee het mogelijk is tekst files van de commodore 128 over te zetten naar een MS-DOS computer. Uiterst nuttig voor de geluksvogels onder ons die beide machines thuis hebben staan, maar ook interresant voor die gene die niet over deze twee computers beschikt. Immers als je zelf niet over een printer beschikt maar de buurman heeft er wel eentje op zijn PC aan gesloten en er moet een tekst uitgeprint worden, no problem, de 128 tekst file over zetten naar een MS-DOS diskette en uitprinten maar. Maar zover is het nog niet, eerst moeten we nog even uitleggen hoe het een en ander werkt.

DE USER 0 functie.

Vrijwel iedere C-128 gebruiker weet dat zijn 1570 en of zijn 1571 diskdrive over een 1541 mode beschikt en als u dat nog niet wist dan weet u het nu dus wel. Het omschakelen van de 1571 mode naar 1541 mode kan op twee manieren gedaan worden. De ene methode is hardware matig, als u overschakeld naar C-64 mode wordt de diskdrive automatisch in de 1541 mode gezet. Maar het over schakelen kan ook software matig en wel als volgt:

```
open 15,8,15,"u0>m0":dclose
```

Dit is nu een voorbeeld van een USER 0 functie. Dit is wel de meest bekende maar er zijn er meer leest u maar verder.
De USER 0 functies werken alleen in de 1570 of de 1571 mode behalve de instructie H en moet aan de volgende systax voldoen.

```
"u0"+chr$(31)+"functie" of
    "u0>functie"
```

functie:

- M1 schakeld de 1571 mode in. De diskdrive elektronica wordt nu omgeschakeld naar 2 MHz. Via deze functie kan de 1571 modus ook in c-64 mode gebruikt worden.

- M0 schakeld de 1541 mode in. De diskdrive werkt nu op 1 MHz.

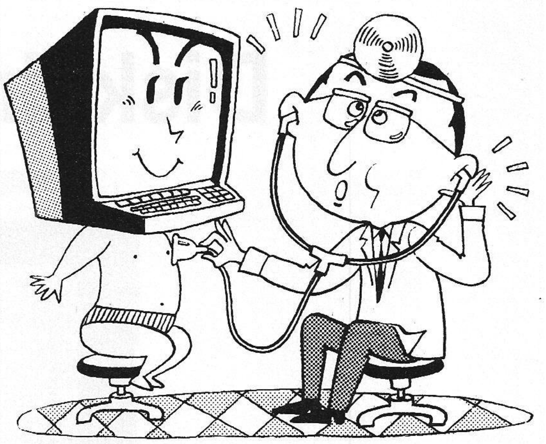
De volgende functie werkt alleen bij de 1571 in de 1541 mode.

- H0 De lees-, en schrijfkop aan kant 1 is nu in gebruik.

- H1 De lees-, en schrijfkop aan kant 2 is nu in gebruik.

Bij de M en H functie kan met bit 7 aangegeven worden of de diskette ook geinitialiseerd moet worden.

0 = de diskette wordt geinitialiseerd.



1 = de diskette wordt niet geinitialiseerd.

Dus met

```
"u0"+chr$(31)+"fucntie"
```

wordt de diskette geïnitialiseerd.

En met

```
"u0"+chr$(159)+"functie"
```

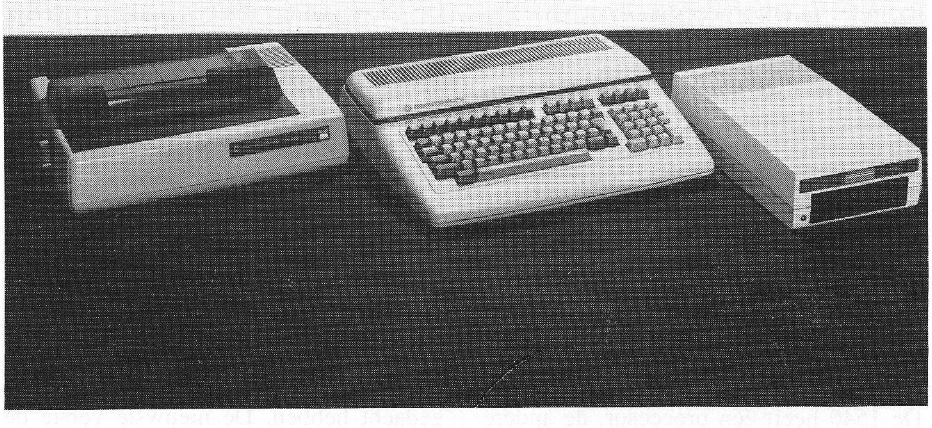
wordt de diskette niet geïnitialiseerd.

*R*x stelt het aantal leespogingen van de drive in, x is hier de ascii waarde van het aantal leespogingen die gewenst zijn.

T met deze instructie kan een ROM checksum routine aangeroepen worden. Deze routine controleert over de ROM nog in een goede staat verkeerd.

X de ascii waarde x wordt nu het nieuwe apparaatadres.

Dit zijn allemaal USER 0 functies die in de normale 1570 of 1571 mode werken. Maar de belangrijkste USER 0 functies gelden als de drive in de CP/M mode wordt gebruikt. CP/M mode? Jazeker de diskdrive beschikt net als de computer over de mogelijkheid om in een CP/M mode te werken. Hiervoor is er in de diskdrive een speciale controller opgenomen, de WD 1770. Met behulp van deze controller kan de 1570 en de 1571 maarliefst 9 verschillende diskette formaten lezen en schrijven. Het programma dat in het volgende nummer van dit blad zal staan werkt met behulp van deze controller. We zullen dan ook uitgebreid op de mogelijkheden van deze speciale drive functies ingaan.



Checksum C-64

Syntax Checksum

Het overtikken van een listing kan een heel karwei zijn en als u een beetje normaal mens bent, dan maakt u daarin beslist een aantal fouten. Nu is niets moeilijker da fouten uit je eigen werk te halen. Al geruime tijd heeft Jan Bodzinga hiervoor een zgn. Checksum-programma geschreven. Om de vele nieuwe lezers van Commodore-info te helpen volgt hieronder nog een keer een volledige uitleg over de werking van dit programma, waarmee het, hoe vreemd dat misschien ook lijkt, echt mogelijk is om met behulp van dit programma de fouten in elke door ons geplaatste listing op te sporen.

Hiervoor gaat u als volgt te werk:

1. U tikt de listing heel zorgvuldig over en SAVEt hem voordat u het programma RUNt op een diskette of cassette.
2. U tikt het RUN commando in. Mocht het programma de boodschap 'FOUT in dataregels!' geven dan heeft u een fout bij het overtikken gemaakt. Herstel dan de fout en SAVE de verbeterde versie. Mocht het programma met de boodschap 'data is weggezet checksum testen met sys...'komen dan is tot dusver alles goed. Het programma is nu in een stukje machinetaalgeheugen gezet. Als u het NEW-commando geeft, blijft het toch in de computer staan.

Alle door ons geplaatste programma's zijn in Basic geschreven.

Als u een programma heeft overgetikt, SAVE het eerst; mocht er iets mis gaan dan hoeft u niet de gehele listing opnieuw te gaan intikken. Als u nu een programma op fouten wilt gaan controleren, kunt u dat in het geheugen laden(wel eerst het checksum programma hebben gerund). Vervolgens typt u zonder het programma te runnen de opdracht sys 49152(c-64) of sys 1536 (c-16 en plus/4) in.

Als alles goed is gegaan loopt er nu een rij regelnummers over het scherm met getallen erachter. Dezelfde lijst staat ook achter elk door ons geplaatste programma. Wijkt nu een nummer achter een regelnummer af van het nummer dat in het blad staat dan heeft u in die regel iets anders ingetikt dan er in het blad stond. U kunt de stroom getallen met de RUN/STOP toets pauzeren en weer vervolgen met de F1 of F7 toets. Het is uitermate belangrijk dat u goed met dit programma overweg kunt en mocht u het niet goed werkend krijgen bel dan gerust even met onze listingservice telefoonlijn. (Maandag 17.00 - 21.00 uur. Telefoonnummer 02155-25162.)

De laatste tijd wordt er weer veel gebeld zodat U nogwel eens in gesprek krijgt, daarom houdt uw vraag kort, vermeld in welk blad het desbetreffende artikel stond. Heeft U veel vragen, of is uw vraag erg uitgebreid, doe het dan schriftelijk, zodat we veel mensen op de maandag avond te woord kunnen staan.

```
1    rem ******************************
2    rem basic loader "SYNTAX.CHECKSUM"
3    rem na de commando's "run" en "new"
4    rem blijft dit programma in het ge-
5    rem heugen. laad het te testen pro-
6    rem gramma en tik daarna sys 49152.
7    rem ******************************
10   i=49152  :rem beginadres
20   reada:ifa<0then40:rem data ingelezen
30   pokei,a:i=i+1:b=b+a:goto20
40   if b<>16844thenprint"[SHIFT-CLR]fo
     ut[SPACE]in[SPACE]dataregels!":b=0:end
50   poke49184,148:poke49185,192
55   i=49300
60   read a: ifa<0then80
70   pokei,a:b=a+b:i=i+1:goto60
80   if b<>20068thenprint"[SHIFT-CLR]fo
     ut[SPACE]in[SPACE]dataregels![SPAC
     E](vanaf[SPACE]regel[SPACE]240)":b
     =0:end
90   print"data[SPACE]is[SPACE]weggezet"
95   print"checksum[SPACE]testen[SPACE]
     met[SPACE]sys49152"
100  data 165,43,166,44,133,163,134,164
     ,169,147
110  data 32,210,255,160,0,240,3,32,73,192
120  data 32,73,192,208,1,96,32,225,255,208
130  data 3,76,116,164,32,81,192,32,73,192
140  data 240,12,201,32,240,247,24,101,
     167,133
150  data 167,76,37,192,166,167,169,0,1
     32,168
160  data 32,205,189,169,13,32,210,255,
     164,168
170  data 76,17,192,200,208,2,230,164,1
     77,163
180  data 96,162,0,189,123,192,240,6,32,210
190  data 255,232,208,245,32,73,192,170
     ,32,73
200  data 192,132,168,32,205,189,162,3,
     169,32
210  data 32,210,255,202,208,250,169,0,
     133,167
220  data 164,168,96,82,69,71,69,76,32,0
230  data  -1
240  data 165,197,201,3,240,7,201,4,240
250  data 6,76,148,192,76,34,192,169
260  data 147,32,210,255,76,161,192
270  data -1
```

** EINDE LISTING checksum 64 **

Checksum Checksum C-64

REGEL 1	249	REGEL 100	183
REGEL 2	84	REGEL 110	158
REGEL 3	105	REGEL 120	232
REGEL 4	2	REGEL 130	183
REGEL 5	246	REGEL 140	96
REGEL 6	152	REGEL 150	96
REGEL 7	249	REGEL 160	127
REGEL 10	157	REGEL 170	71
REGEL 20	64	REGEL 180	223
REGEL 30	38	REGEL 190	73
REGEL 40	57	REGEL 200	79
REGEL 50	14	REGEL 210	109
REGEL 55	251	REGEL 220	106
REGEL 60	192	REGEL 230	225
REGEL 70	42	REGEL 240	16
REGEL 80	244	REGEL 250	163
REGEL 90	245	REGEL 260	92
REGEL 95	237	REGEL 270	22

PRINT OUT C-64 met o.a.Spreadsheet

Apenspel 64

Dit spel, geschreven door F. Reijsbergen, moet tegen de computer worden gespeeld. Onze trouwe lezers zullen zeker uit de naam van de inzender kunnen op maken wat dit betekent, juist ja, niet eenvoudig om het spel te kunnen winnen. Op het scherm worden een aantal apen getekent, en deze zijn ziek. De bedoeling is nu om deze apen beter te maken. De computer probeert dit ook. Maar de winnaar is diegenen die de **laatste** aap beter maakt. Er mogen maximaal twee apen tegelijk worden aangewezen, en alleen nog als ze vlak naast elkaar staan.

```
1       rem het apen spel / commodore 64
2       rem door fons reijsbergen
3       rem      leidschendam
4       rem
10      poke53280,5:poke53281,5:print"[CTR
        L 1][SHIFT-CLR]":gosub520
20      l$="[CTRL-8]#%%[CRSR-DOWN][3xCRSR-
        LEFT](’&[CRSR-DOWN][3xCRSR-LEFT])*
        +[2xCRSR-UP][CRSR-RIGHT][CTRL-1]":
        k$="#$%[CRSR-DOWN][3xCRSR-LEFT]&’(
        [CRSR-DOWN][3xCRSR-LEFT])*+[2xCRSR
        -UP][CRSR-RIGHT]":t$="[4xCRSR-DOWN
        ]":k=7:restore
30      print"[SHIFT-CLR]":a$="11111111111
        11":dimt(13):fori=1to13:readt(i):n
        ext:e=1
40      gosub350:input"wie[SPACE]mag[SPACE
        ]er[SPACE]beginnen[SPACE]comp[SPAC
        E]of[SPACE]mens";i$:ifleft$(i$,1)=
        "c"then220
50      e=0
60      i$="":input"[CRSR-UP]welke[SPACE]a
        ap[SPACE]maak[SPACE]je[SPACE]beter
        [15xSPACE][8xCRSR-LEFT]";i$:ifi$="
        "then60
70      iflen(i$)>2then60
80      l=len(i$):fori=1tol:ifmid$(i$,i,1)
        <"a"ormid$(i$,i,1)>"m"then60
90      nexti:a=asc(i$)-64:a$=mid$(a$,1,a-
        1)+"0"+right$(a$,13-a)
100     ifl=2anda<13thena$=mid$(a$,1,a)+"0
        "+right$(a$,12-a)
110     w=1:gosub450:gosub350:print"een[SP
        ACE]ogenblik[SPACE]ik[SPACE]denk[S
        PACE]na[12xSPACE]"
120     ife=1then670
130     fork=1to13:ifmid$(a$,k,1)="0"then1
        90
140     b$=mid$(a$,1,k-1)+"0"+right$(a$,13
        -k):gosub280
150     gosub320:ifok=1thend=0:goto230
160     ifk=13ormid$(a$,k+1,1)="0"then190
170     b$=mid$(b$,1,k)+"0"+right$(b$,12-k
        ):gosub280:ifa=1andr(1)=0thend=1:g
        oto230
180     gosub320:ifok=1thend=1:goto230
190     nextk:fori=1to13:ifmid$(a$,k,1)="1
        "then220
200     fork=1to13:ifmid$(a$,k,1)="1"then2
        20
210     nextk:w=0:goto450
220     b$=mid$(a$,1,k-1)+"0"+right$(a$,13
        -k):d=0
230     a$=b$:gosub350:print"ik[SPACE]neem
        [SPACE]aapje[SPACE]";chr$(k+64);:i
        fd=1thenprintchr$(k+65);
240     print"[23xSPACE]"
250     w=0:gosub450
260     geti$:ifi$=""then260
270     goto60
280     fori=1to10:r(i)=0:next:a=1:fori=1t
        o13
290     ifmid$(b$,i,1)="1"thenr(a)=r(a)+1:
        goto310
300     ifr(a)<>0thena=a+1
310     next:return
320     b=0:fort=1toa:b=b+t(r(t)):next:c$=
        mid$(str$(b),2):ok=0
330     fori=1tolen(c$):ifval(mid$(c$,i,1)
        )/2<>int(val(mid$(c$,i,1))/2)thenr
        eturn
340     nexti:oke=1:return
350     print"[HOME]************[SPACE]het
        [SPACE]apen[SPACE]spel[SPACE]*****
        ********"
360     print"[3xCRSR-DOWN]"tab(11)"a[3xSP
        ACE]b[3xSPACE]c[3xSPACE]d[3xSPACE]
        e":printt$tab(13)"f[3xSPACE]g[3xSP
        ACE]h[3xSPACE]i"
370     printt$tab(15)"j[3xSPACE]k[3xSPACE
        ]l":printt$tab(19)"m[HOME]":printt
        ab(10);:fori=1to5
380     gosub430:nexti:print:printt$tab(12
        );:fori=6to9:gosub430:next:print
390     printt$tab(14);:fori=10to12:gosub4
        30:next:print:printt$tab(18);
400     ifmid$(a$,13,i)="1"thenprintl$:got
        o420
410     printk$
420     print"[3xCRSR-DOWN]":return
430     ifmid$(a$,i,1)="1"thenprintl$;:ret
        urn
440     printk$;:return
450     ifa$<>"0000000000000"thenreturn
460     gosub350:ifw=1thenprint"de[SPACE]m
        ens[SPACE]";:goto480
470     print"de[SPACE]computer[SPACE]";
480     print"heeft[SPACE]gewonnen[17xSPAC
        E]"
490     input"nog[SPACE]maals[SPACE]een[SP
        ACE]spel[SPACE]j/n[SPACE]";i$
500     ifi$="j"thenrun20
510     stop
520     print"[HOME]************[SPACE]het
        [SPACE]apen[SPACE]spel[SPACE]*****
        ********[2xCRSR-DOWN]"
530     print"maak[SPACE]de[SPACE]apen[SPA
        CE]beter.":print"[CRSR-DOWN]diegen
        e[SPACE]die[SPACE]de[SPACE]laatste
        [SPACE]aap[SPACE]beter[SPACE]maakt
        "
540     print"heeft[SPACE]gewonnen.":print
        "[2xCRSR-DOWN]je[SPACE]maakt[SPACE
        ]een[SPACE]aap[SPACE]beter[SPACE]d
        oor[SPACE]de[SPACE]letter"
550     print"in[SPACE]te[SPACE]typen[SPAC
        E]die[SPACE]onder[SPACE]de[SPACE]a
        ap[SPACE]staat."
560     print"je[SPACE]kunt[SPACE]een[SPAC
        E]of[SPACE]twee[SPACE]apen[SPACE]b
```

```
     eter[SPACE]maken.[3xSPACE]als[SPAC
     E]je[SPACE]kiest[SPACE]voor[SPACE]
     twee[SPACE]apen";
570  print"[SPACE]dan[SPACE]moeten[2xSP
     ACE]dat[SPACE]apen[SPACE]zijn[SPAC
     E]die[SPACE]naast[SPACE]elkaar[SPA
     CE]staan."
580  print"bv.[SPACE]-ab-[SPACE]of[SPAC
     E]-ef-.[CRSR-DOWN]":print"als[SPAC
     E]de[SPACE]computer[SPACE]geweest[
     SPACE]is[SPACE]moet[SPACE]je[SPACE
     ]op"
590  print"een[SPACE]toets[SPACE]drukke
     n[SPACE]en[SPACE]dan[SPACE]pas[SPA
     CE]je[SPACE]keus[SPACE]in-typen."
600  fori=1to13:reada:next
610  poke56334,peek(56334)and254:poke1,
     peek(1)and251:fori=0to500
620  pokei+12288,peek(i+53248):next:for
     i=280to351:reada:pokei+12288,a:nex
     t
630  poke1,peek(1)or4:poke56334,peek(56
     334)or1:poke53272,28
640  print"[2xCRSR-DOWN][9xSPACE]druk[S
     PACE]op[SPACE]een[SPACE]toets"
650  geti$:ifi$=""then650
660  return
670  d=0:ifl=2thend=1
680  ifa>7thenk=a-7
690  ifa<7thenk=a+7
700  b$=mid$(a$,1,k-1)+"0"+right$(a$,13
     -k)
710  ifl=2andk<13thenb$=mid$(b$,1,k)+"0
     "+right$(b$,12-k)
720  goto230
730  data1,10,11,1,100,11,10,1,100,10,1
     10,100,1,0,0,0,0,1,3,103,158,8,50,
     127
740  data255,255,220,8,107,0,0,0,128,19
     2,224,118,57,206,174,174,143,95,39
     ,7,7
750  data235,235,73,0,128,227,182,42,18
     7,187,57,121,250,244,240,112,14,12
     ,7,3,1
760  data0,0,0,34,0,28,34,193,34,28,0,5
     6,24,112,96,192,0,0,0
```

EINDE LISTING apenspel 64

Checksum apenspel 64

REGEL 1	6	REGEL 390	67
REGEL 2	41	REGEL 400	64
REGEL 3	36	REGEL 410	8
REGEL 4	143	REGEL 420	216
REGEL 10	199	REGEL 430	183
REGEL 20	18	REGEL 440	11
REGEL 30	7	REGEL 450	61
REGEL 40	122	REGEL 460	254
REGEL 50	39	REGEL 470	16
REGEL 60	33	REGEL 480	170
REGEL 70	252	REGEL 490	150
REGEL 80	6	REGEL 500	203
REGEL 90	203	REGEL 510	144
REGEL 100	140	REGEL 520	101
REGEL 110	120	REGEL 530	120
REGEL 120	247	REGEL 540	183
REGEL 130	55	REGEL 550	237
REGEL 140	118	REGEL 560	73
REGEL 150	137	REGEL 570	78
REGEL 160	50	REGEL 580	59
REGEL 170	15	REGEL 590	235
REGEL 180	138	REGEL 600	115
REGEL 190	55	REGEL 610	2
REGEL 200	50	REGEL 620	4
REGEL 210	156	REGEL 630	129
REGEL 220	117	REGEL 640	59
REGEL 230	217	REGEL 650	120
REGEL 240	221	REGEL 660	142
REGEL 250	153	REGEL 670	233
REGEL 260	117	REGEL 680	123
REGEL 270	239	REGEL 690	124
REGEL 280	195	REGEL 700	21
REGEL 290	88	REGEL 710	173
REGEL 300	185	REGEL 720	30
REGEL 310	74	REGEL 730	244
REGEL 320	70	REGEL 740	164
REGEL 330	194	REGEL 750	246
REGEL 340	143	REGEL 760	49
REGEL 350	67		
REGEL 360	174		
REGEL 370	136		
REGEL 380	103		

Galgje 64

Ook dit overbekende spel is er weer één uit de grote voorraad van F. Reijsbergen uit Leidschendam. Een uitleg hoeven we er eigenlijk niet meer bij te geven, de bedoeling is het woord dat de computer in zijn geheugen heeft genomen te raden. Elke keer dat er op een verkeerde letter wordt gegokt, dan kom je een stukje dichter bij de galg. En elke stap brengt je, gelukkig alleen maat figuurlijk, dichter bij het einde.

```
1    rem galgje / cbm-64
2    rem program door fons reijsbergen
3    rem sprites door fred reijsbergen
5    rem       leidschendam
6    rem
10   poke52,48:poke56,48:clr:poke53269,
     0:poke53280,3:poke53281,3
20   gosub400
30   t$="[HOME][22xCRSR-DOWN]":p$="."
40   l$="[34xSHIFT-*]":g0=0:f0=0
50   pokev+21,0:print"[SHIFT-CLR][HOME]
     [CTRL-1][CRSR-DOWN]"tab(16)"GALGJE
     "
60   print"[COM-2]"left$(t$,13)tab(16)"
     [CTRL-9][SPACE][CTRL-0][COM-I]"
70   print"[16xCOM-`][CTRL-9][3xSPACE][
     CTRL-0][COM-I][20xCOM-`][CTRL-2]"
80   print"[CRSR-DOWN][COM-A][2xSHIFT-*
     ]"l$"[2xSHIFT-*][COM-S][SHIFT--]":
     print"[CRSR-LEFT][2xSHIFT--][2xSPA
```

```
    CE]Het[SPACE]woord:[26xSPACE][2xSH
    IFT -]"
90  print"[CRSR-LEFT][2xSHIFT--][11xSP
    ACE]Geef[SPACE]letter[5xSPACE]":pr
    int"[CRSR-LEFT][2xSHIFT--][2xSPACE
    ]"l$"[2xSPACE][SHIFT--]";
100 print"[SHIFT--][2xSPACE]Goed[8xSPA
    CE]Fout[8xSPACE]Score[4xSPACE]%[2x
    SPACE][SHIFT--]";:print"[COM-Z][2x
    SHIFT-*]"l$"[2xSHIFT-*][COM-X]";
110 poke2040,192:ps=0:gosub340:pokev+2
    1,63:ifg0+f0>0thengosub330
120 w$=a$(rnd(0)*an+1):r$="":le=len(w$
    ):fori=1tole:r$=r$+".":next:ps=1:r
    l=1
130 gosub340:foril=1to3:gosub310:nexti
    l
140 printleft$(t$,21)tab(24)rlleft$(t$
    ,19)tab(14)r$:rl=rl+1:poke198,0
150 geti$:ifasc(i$+chr$(0))<65orasc(i$
    +chr$(0))>90then150
160 gosub310:k=0:fori=1tole
170 ifi$=mid$(w$,i,1)andmid$(r$,i,1)=p
    $thenr$=left$(r$,i-1)+i$+mid$(r$,i
    +1):k=1
180 next:ifr$=w$thenprintleft$(t$,19)t
    ab(14)w$:gosub220:goto50
190 ifk=1then140
200 ps=ps+1:ifps<11thengosub340:goto14
    0
210 printleft$(t$,19)tab(14)w$:gosub36
    0:goto50
220 ps=12:gosub340:g0=g0+1:gosub330:pr
    intleft$(t$,21)tab(12)"Nog[SPACE]e
    en[SPACE]woord[SPACE]??"
230 fori=125to122step-1:pokev+7,i:poke
    v+9,i:forj=1to10:nextj,i
240 geti$:ifi$="j"thenreturn
250 fori=122to125:pokev+7,i:pokev+9,i:
    forj=1to10:nextj,i
260 geti$:ifi$="j"thenreturn
270 goto230
280 s=54272:fori=0to23:pokes+i,0:next:
    pokes+24,15:return
290 gosub280:pokes+5,10:pokes+6,75:pok
    es+15,10:pokes+14,10:pokes+1,4:pok
    es+4,21
300 fori=1to50:next:pokes+4,20:return
310 gosub280:pokes+6,240:pokes,15:poke
    s+1,67:pokes+4,17
320 forpz=1to100:next:pokes+4,16:retur
    n
330 printt$tab(8)g0;tab(20)f0;tab(33)i
    nt(g0*(100/(g0+f0)+.1))"[CRSR-LEFT
    ]%[SPACE]":return
340 p1=a(ps,1):p2=a(ps,2):pokev+2,p1:p
    okev+3,p2:pokev+4,p1:pokev+5,p2
350 pokev+6,p1-1:pokev+7,p2-15:pokev+8
    ,p1-1:pokev+9,p2-15:return
360 poke2040,193:fori=a(10,2)toa(11,2)
    :pokev+3,i:pokev+5,i:forj=1to10:ne
    xtj,i
370 f0=f0+1:gosub330:printleft$(t$,21)
    tab(12)"Nog[SPACE]een[SPACE]woord[
    SPACE]??"
380 gosub290:fori=1to300:geti$:ifi$="j
    "thenreturn
390 next:goto380
400 printchr$(14)"[SHIFT-CLR][CTRL-2][
    6xCRSR-DOWN][14xCRSR-RIGHT]wacht[S
    PACE]eventjes"
410 v=12352:fori=0to62:reada:poke12288
    +i,a:pokev+i,a:next
420 fori=v+25tov+53step3:pokei,192:nex
    t:fori=12416to12734:reada:pokei,a:
    next
430 v=194:fori=2041to2045:pokei,v:v=v+
    1:next:v=53248:dima(12,2)
440 fori=v+37tov+44:reada:pokei,a:next
    :fori=0to12:reada(i,1),a(i,2):next
450 pokev+23,32:pokev+27,0:pokev+28,21
    :pokev+29,1
460 pokev,105:pokev+1,140:pokev+10,107
    :pokev+11,98
470 readan:dima$(an):fori=1toan:reada$
    (i):next
480 print"[SHIFT-CLR]":return
490 data32,0,0,32,0,0,32,0,0,32,0,0,32
    ,0,0,32,0,0,170,170,170,170,170,17
    0
500 data34,255,136,42,255,168,40,0,40,
    32,0,8,32,0,8,32,0,8,32,0,8,32,0,8
    ,32,0
510 data8,32,0,8,32,0,8,32,0,8,32,0,8
520 data1,254,0,7,3,128,4,132,128,4,13
    2,128,4,132,128,2,133,0,2,133,0,2,
    133,0
530 data3,135,0,4,252,128,9,182,64,7,5
    1,128,1,50,0,1,50,0,1,50,0,1,50,0
540 data1,50,0,1,50,0,0,180,0,1,122,0,
    2,73,0,0
550 data0,252,0,3,255,0,3,255,0,3,255,
    0,3,255,0,3,255,0,3,255,0,3,255,0,
    3,255
560 data0,1,169,0,5,169,64,0,168,0,0,1
    68,0,0,168,0,0,168,0,0,168,0,0,168
    ,0,0
570 data168,0,0,168,0,0,68,0,1,69,0,0
580 data0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,195,0,0,36,0,0,24
590 data0,0,0,0,1,60,128,1,66,128,0,12
    9,0,0,102,0,0,36,0,0,36,0,0,36,0,0
    ,36
600 data0,0,36,0,0,36,0,0
610 data0,0,0,0,0,0,0,0,0,0,0,0,0,170,
    0,2,170,128,2,170,128,2,85,128,2,2
    15
620 data128,2,85,128,2,85,128,1,85,64,
    0,125,0,0,85,0,0,20,0,0,20,0,0,20
630 data0,0,20,0,0,20,0,0,20,0,0,20,0,
    0
640 data127,255,255,24,192,4,25,128,4,
    27,0,4,30,0,4,28,0,4,24,0,4,24,0,4
650 data24,0,4,24,0,4,24,0,10,24,0,17,
    24,0,17,24,0,14,24,0,0,24,0,0,24,0
    ,0
660 data24,0,0,24,0,0,24,0,0,24,0,0
670 data10,6,9,0,7,0,9,9
680 data15,15,255,140,238,140,225,140,
    210,140,195,140,180,140,170,137,15
    5,129
690 data135,125,118,125,118,131,50,140
700 :
710 :
720 rem********************************
                   *de woordenlijst kun je l
```

```
         anger*
730      rem*maken. het getal in regel 770*
                   *geeft het aantal woorden
          aan *
740      rem******************************
750      :
760      :
770      data58:rem****aantal woorden****
780      datagalgje,auto,boot,vliegtuig,fie
         ts,bromfiets,printer,computer
790      datadiskdrive,disk,beeldscherm,toe
         tsenbord,spel,prgramma,potlood,pap
         ier
800      datawoordenboek,woorden,boek,leesb
         oek,plant,onkruid,bloem,bloembol
810      datavogel,vis,zoogdier,mens,insekt
         ,lucht,aarde,vuur,zon,maan,sterren
820      dataland,zee,strand,zand,water,woe
         stijn,oerwoud,moeras,droog,nat,war
         m
830      datakoud,hoog,laag,ver,dichtbij,hi
         er,daar,waar,heen,terug,komen,gaan
```

EINDE LISTING galgje 64

Checksum Galgje 64

REGEL 1	209	REGEL 410	114
REGEL 2	65	REGEL 420	7
REGEL 3	62	REGEL 430	184
REGEL 5	36	REGEL 440	62
REGEL 6	143	REGEL 450	134
REGEL 10	219	REGEL 460	56
REGEL 20	33	REGEL 470	177
REGEL 30	201	REGEL 480	56
REGEL 40	11	REGEL 490	165
REGEL 50	213	REGEL 500	155
REGEL 60	80	REGEL 510	255
REGEL 70	56	REGEL 520	146
REGEL 80	43	REGEL 530	105
REGEL 90	213	REGEL 540	142
REGEL 100	192	REGEL 550	144
REGEL 110	18	REGEL 560	105
REGEL 120	46	REGEL 570	3
REGEL 130	164	REGEL 580	161
REGEL 140	41	REGEL 590	8
REGEL 150	228	REGEL 600	169
REGEL 160	164	REGEL 610	213
REGEL 170	3	REGEL 620	105
REGEL 180	128	REGEL 630	39
REGEL 190	245	REGEL 640	171
REGEL 200	115	REGEL 650	207
REGEL 210	244	REGEL 660	127
REGEL 220	211	REGEL 670	144
REGEL 230	54	REGEL 680	192
REGEL 240	181	REGEL 690	67
REGEL 250	177	REGEL 700	58
REGEL 260	181	REGEL 710	58
REGEL 270	30	REGEL 720	92
REGEL 280	181	REGEL 730	174
REGEL 290	196	REGEL 740	165
REGEL 300	202	REGEL 750	58
REGEL 310	209	REGEL 760	58
REGEL 320	92	REGEL 770	216
REGEL 330	79	REGEL 780	55
REGEL 340	142	REGEL 790	91
REGEL 350	64	REGEL 800	213
REGEL 360	127	REGEL 810	116
REGEL 370	129	REGEL 820	174
REGEL 380	53	REGEL 830	173
REGEL 390	224		
REGEL 400	201		

Spreadsheet 64

De Heer O.R. Vos uit Amersfoort volgde een cursus Lotus 1 2 3. Thuis achter zijn trouwe Commodore 64 heeft hij zijn pas verworven kennis gekombineerd met reeds bestaande programmeer kennis gekoppeld tot het volgende programma. Een uitleg hoe het programma gebruikt moet worden is in het programma opgenomen. Deze wordt verkregen door het indrukken van de ?-toets.

```
5       print"[SHIFT-CLR]":poke53280,0:pok
        e53281,0:dimcm$(2600):poke650,0
10      rem
15      l1$="[CTRL-9][4xSPACE][CTRL-0][10x
        SHIFT-*][SHIFT-+][10xSHIFT-*][SHIF
        T +][10xSHIFT-*][COM-W][COM-G]"
20      l2$="[4xCRSR-RIGHT][10xSPACE]B[10x
        SPACE]B[10xSPACE]B[COM-G]"
25      w$="[10xSPACE][SHIFT--][11xCRSR-LE
        FT][CTRL-4]"
30      x$="[CTRL-9][37xSPACE][CTRL-0][COM
        G]"
35      y$="[39xSPACE]"
40      z$="[CTRL-9]abcdefghijklmnopqrstuv
        wxyz[CTRL-0]"
45      r1=1:kl=2:r2=1:cd=1
50      print"[HOME][4xCRSR-DOWN][CTRL-6]"
        ;
55      print"[CTRL-9][9xSPACE]"mid$(z$,kl
        ,1)"[9xSPACE]"mid$(z$,kl+1,1);
60      print"[11xSPACE]"mid$(z$,kl+2,1)"[
        5xSPACE][CTRL-0][COM-H]":printl1$:
        d=r2
65      fori=r1tor1+7
70      print"[CTRL-9][4xSPACE][4xCRSR-LEF
        T][CTRL-6]"i"[CTRL-0]":print"[CRSR
        -UP][4xCRSR-RIGHT]"w$;left$(cm$(d)
        ,10)"[CTRL-6]";
75      printtab(15)w$;left$(cm$(d+100),10
        )"[CTRL-6]";
80      printtab(26)w$;left$(cm$(d+200),10
        )"[CTRL-6]"tab(37)"[COM-H]"
85      d=d+1:printl1$:next
90      print"[CTRL-9][6xSPACE](c)[SPACE]1
        987[2xSPACE]o.r.vos[SPACE]-[SPACE]
        software[3xSPACE][CTRL-0][COM-G]"
95      printtab(14)"[CTRL-9][CTRL-4][PIJL
        LINKS][SPACE]=[SPACE]escape[CTRL-6
        ][CTRL-0][3xSPACE]?[SPACE]=[SPACE]
        help[HOME]"
100     print"[HOME][CTRL-0][SPACE][CTRL-4
        ]commando's[SPACE][CTRL-9][SPACE]n
        [SPACE]g[SPACE]h[SPACE]w[SPACE]f[S
        PACE]c[SPACE]s[SPACE]l[SPACE]z[SPA
        CE]m[SPACE][CTRL-6][CTRL-0][SPACE]
        ";y$
105     rem
110     printy$"[CRSR-UP][CTRL-6][SPACE]co
        mmand:";
115     gosub480
120     ifa$="n"then920
125     ifa$="g"thent=1:fo=0:goto195
130     ifa$="h"then45
135     ifa$="w"then510
140     ifa$="f"then310
```

```
145   ifa$="c"then525
150   ifa$="m"then520
155   ifa$="s"then665
160   ifa$="l"then540
165   ifa$="z"then970
170   ifa$="[CRSR-RIGHT]"andkl<25thenkl=
      kl+1:r2=r2+100:goto50
175   ifa$="[CRSR-LEFT]"andkl>2thenkl=kl
      -1:r2=r2-100:goto50
180   ifa$="[CRSR-DOWN]"andrl<93thenrl=r
      l+1:r2=r2+1:goto50
185   ifa$="[CRSR-UP]"andrl>1thenrl=rl-1
      :r2=r2-1:goto50
190   goto115
195   print"[HOME][2xCRSR-DOWN][CTRL-6]g
      oto[SPACE]cel:";
200   c=0:r=0:o=0:v$=""
205   gosub480
210   ifa$=chr$(13)then205
215   c=asc(a$)-65:ifc=0then225
220   r=100
225   ifc<0orc>26then200
230   ift=1thent=0:goto240
235   return
240   printa$;
245   gosub480
250   ifa$=chr$(13)then275
255   b=asc(a$):ifb<48orb>57then245
260   v$=v$+a$:o=val(v$):ifo<0oro>100the
      nprint"[CTRL-0][3xCRSR-LEFT][3xSPA
      CE][3xCRSR-LEFT]";:t=1:goto200
265   printa$;
270   goto245
275   cel=(c*r)+o:ifcel>cdthencd=cel
280   iffo=1thenreturn
285   print"[HOME][2xCRSR-DOWN][CTRL-9]e
      nter[CTRL-0][8xSPACE][7xCRSR-LEFT]
      ";:poke19,1:inputcm$(cel):poke19,0
290   cm$(cel)=cm$(cel)+"[10xSPACE]"
295   goto50
300   print"[HOME][CTRL-9][CTRL-2][2xSPA
      CE]laatst[SPACE]ingevoerde[SPACE]c
      el[SPACE]is[SPACE]leeg[6xSPACE][CT
      RL 0][CTRL-6]":gosub710:goto310
305   print"[HOME][CTRL-9][CTRL-2][2xSPA
      CE]cel[SPACE]bevat[SPACE][CTRL-3]g
      een[CTRL-2][SPACE]rekenkundig[SPAC
      E]getal[3xSPACE][CTRL-0][CTRL-6]":
      gosub710
310   print"[HOME][CTRL-9][CTRL-4]u[SPAC
      E]kunt[SPACE]alleen[3xSPACE]+[SPAC
      E]-[SPACE]*[SPACE]~[SPACE]/[3xSPAC
      E]gebruiken[CTRL-0][CTRL-6][15xSPA
      CE]"
315   print"[HOME][2xCRSR-DOWN][CTRL-9]f
      ormule:[CTRL-0][10xSPACE][10xCRSR-
      LEFT]";:q=0:uit=0:do=0:t=0
320   gosub200:printa$;
325   gosub480
330   ifa$="+"thenprinta$;:goto380
335   ifa$="-"thenprinta$;:goto380
340   ifa$="*"thenprinta$;:goto380
345   ifa$="/"thenprinta$;:goto380
350   ifa$="~"thenprinta$;:goto380
355   ifa$="="thenprinta$;:uit=1:goto380
360   b=asc(a$):ifb<48orb>57then325
365   v$=v$+a$:o=val(v$):ifo<0oro>100the
      nprint"[CTRL-0][3xCRSR-LEFT][3xSPA
      CE][3xCRSR-LEFT]";:fo=1:gosub200
370   printa$;
375   goto325
380   ifdo=1thend$=r$:r$=a$:goto390
385   r$=a$
390   q=q+1
395   cel=(c*r)+o
400   ifcm$(cel)=""then300
405   ifasc(cm$(cel))=<46orasc(cm$(cel))
      =>58then305
410   ifq=1thenf(1)=val(cm$(cel)):v$="":
      do=1:goto320
415   ifq=2thenf(2)=val(cm$(cel)):v$=""
420   ifd$="+"thenuit$=str$(f(1)+f(2))
425   ifd$="-"thenuit$=str$(f(1)-f(2))
430   ifd$="*"thenuit$=str$(f(1)*f(2))
435   ifd$="/"thenuit$=str$(f(1)/f(2))
440   ifd$="[KWADRAATPIJL]"thenuit$=str$
      (f(1)[KWADRAATPIJL]f(2))
445   f(1)=val(uit$):q=1
450   ifuit=1thenuit=0:q=0:goto460
455   goto320
460   fo=1:t=1:gosub200:fo=0:t=0
465   cm$(cel)=str$(f(1)):f(1)=0
470   ha=val(cm$(cel))
475   cm$(cel)=mid$(str$(int((ha+0.51*10
      [KWADRAATPIJL]-2)*10[KWADRAATPIJL]2
      )*10[KWADRAATPIJL]-2),2,9):goto50
480   poke204,0:poke198,0:wait198,1:geta
      $
485   poke207,0:poke204,1
490   ifa$=chr$(20)then480
495   ifa$="?"then720
500   ifa$="[PIJL LINKS]"then45
505   return
510   print"[HOME]"y$y$y$y$"[HOME][CTRL
      9][CTRL-4][SPACE]wissen[SPACE][CTR
      L 0]"
515   print"welke[SPACE]cel:[CTRL-6]";:f
      o=1:t=1:gosub200:cm$(cel)="":fo=0:
      goto50
520   print"[HOME]"y$y$y$y$"[HOME][CTRL
      9][CTRL-4][SPACE]verplaatsen[SPACE
      ][CTRL-0]":v=1:goto530
525   print"[HOME]"y$y$y$y$"[HOME][CTRL
      9][CTRL-4][SPACE]copy[SPACE][CTRL
      0]"
530   print"welke[SPACE]cel:[CTRL-6]";:f
      o=1:t=1:gosub200:c$=cm$(cel):ifv=1
      thencm$(cel)="":v=0
535   print"[SPACE][CTRL-4]naar[SPACE]ce
      l:[CTRL-6]";:fo=1:t=1:gosub200:cm$
      (cel)=c$:fo=0:t=0:goto50
540   print"[SHIFT-CLR][CTRL-9][CTRL-4][
      SPACE]scanning[SPACE][CTRL-0]"
545 print"file's[SPACE]on[SPACE]disk[S
      PACE]:[CRSR-DOWN][7xCRSR-LEFT][CTR
      L 6]bloks[3xSPACE]prog.naam[CTRL-4
      ]":d$="":a$="":b$="":c$=""
550   open1,8,0,"$0":get#1,a$,b$
555   get#1,a$,b$
560   get#1,a$,b$
565   m=0
570   ifa$<>""thenm=asc(a$)
575   ifb$<>""thenm=m+asc(b$)*256
580   d$=d$+"[CTRL-9][SPACE]"+mid$(str$(
      m),2)+"[SPACE][CTRL-0][SPACE]--[SP
      ACE]"
```

```
585  get#1,b$:ifst<>0thenclose1:print"[
     CRSR-DOWN]terug[SPACE]naar[SPACE]m
     enu[3xSPACE]j/n[SPACE]";:gosub480:
     goto900
590  ifb$<>chr$(34)then585
595  get#1,b$:ifb$<>chr$(34)thend$=d$+b
     $:goto595
600  get#1,b$:ifb$=chr$(32)then600
605  c$=""
610  c$=c$+b$:get#1,b$:ifb$<>""then610
615  ifright$(d$,1)="[PIJL LINKS]"thenp
     rinttab(12)d$
620  d$="":goto555
625  print:print"[CTRL-9][SPACE]laden[S
     PACE][CTRL-0]":input"[CTRL-4]file-
     naam[SPACE]:[CTRL-6]";fl$
630  open2,8,2,"`0:"+fl$+"[SPACE][PIJL
     LINKS],s,r"
635  input#2,cd
640  forcel=1tocd
645  input#2,cm$(cel)
650  ifcm$(cel)="."thencm$(cel)=""
655  next
660  close2:print"[SHIFT-CLR]":goto45
665  print"[HOME]"y$y$y$y$"[HOME][CTRL
     9][CTRL-4][SPACE]saven[SPACE][CTRL
      0]"
670  input"[CTRL-4]file-naam[SPACE]:[CT
     RL 6]";fl$
675  open2,8,2,"`0:"+fl$+"[SPACE][PIJL
     LINKS],s,w"
680  print#2,cd
685  forcel=1tocd
690  ifcm$(cel)=""thenprint#2,".":goto7
     00
695  print#2,cm$(cel)
700  next
705  goto660
710  forw=0to3000:next:return
715  next
720  print"[HOME][4xCRSR-DOWN]"
725  print"[4xCRSR-RIGHT][32xSPACE]"
730  print"[4xCRSR-RIGHT][SPACE]n-toets
     [SPACE]=[SPACE]wis[SPACE]gehele[SP
     ACE]sheet[5xSPACE]"
735  print"[4xCRSR-RIGHT][SPACE]g-toets
     [SPACE]=[SPACE]ga[SPACE]naar[SPACE
     ]cel[SPACE]x[8xSPACE]"
740  print"[4xCRSR-RIGHT][SPACE]h-toets
     [SPACE]=[SPACE]terug[SPACE]naar[SP
     ACE]cel[SPACE]a1[4xSPACE]"
745  print"[4xCRSR-RIGHT][SPACE]w-toets
     [SPACE]=[SPACE]wis[SPACE]een[SPACE
     ]cel[10xSPACE]"
750  print"[4xCRSR-RIGHT][SPACE]m-toets
     [SPACE]=[SPACE]verplaats[SPACE]1[S
     PACE]cel[SPACE]naar[SPACE]"
755  print"[4xCRSR-RIGHT][11xSPACE]een[
     SPACE]andere[SPACE]cel[7xSPACE]"
760  print"[4xCRSR-RIGHT][SPACE]c-toets
     [SPACE]=[SPACE]copieer[SPACE]1[SPA
     CE]cel[SPACE]naar[3xSPACE]"
765  print"[4xCRSR-RIGHT][11xSPACE]een[
     SPACE]andere[SPACE]cel[7xSPACE]"
770  print"[4xCRSR-RIGHT][SPACE]s-toets
     [SPACE]=[SPACE]bewaar[SPACE]sheet[
     9xSPACE]"
775  print"[4xCRSR-RIGHT][SPACE]l-toets
     [SPACE]=[SPACE]laad[SPACE]sheet[11
     xSPACE]"
780  print"[4xCRSR-RIGHT][32xSPACE]"
785  print"[4xCRSR-RIGHT][32xSPACE]"
790  print"[4xCRSR-RIGHT][SPACE]cursor-
     toetsen[SPACE]=[SPACE]wandelen[SPA
     CE]door[SPACE]"
795  print"[4xCRSR-RIGHT][18xSPACE]de[S
     PACE]sheet[6xSPACE]"
800  print"[4xCRSR-RIGHT][2xSPACE][CTRL
      9][CTRL-4][SPACE]druk[3xSPACE]spa
     tie[2xSPACE][CTRL-0][CTRL-6][14xSP
     ACE]"
805  print"[4xCRSR-RIGHT][32xSPACE][HOM
     E][2xCRSR-DOWN][8xCRSR-RIGHT]";
810  gosub480
815  print"[HOME][4xCRSR-DOWN]"
820  print"[4xCRSR-RIGHT][32xSPACE]"
825  print"[4xCRSR-RIGHT][SPACE]z-toets
     [SPACE]=[SPACE]zoek[SPACE]een[SPAC
     E]gegeven[5xSPACE]"
830  print"[4xCRSR-RIGHT][32xSPACE]"
835  print"[4xCRSR-RIGHT][SPACE]f-toets
     [SPACE]=[SPACE]b.v.[SPACE]a1*b4-c2
     +h6/d3=[SPACE]"
840  print"[4xCRSR-RIGHT][11xSPACE]dus[
     SPACE][CTRL-9][CTRL-2]eerst[CTRL-0
     ][CTRL-6][SPACE]formuleren[SPACE]"
845  print"[4xCRSR-RIGHT][11xSPACE]dan[
     SPACE]aangeven[SPACE]in[SPACE]welke"
850  print"[4xCRSR-RIGHT][11xSPACE]cel[
     2xSPACE]de[SPACE]uitkomst[5xSPACE]"
855  print"[4xCRSR-RIGHT][11xSPACE]komt
     [SPACE]te[SPACE]staan[8xSPACE]"
860  print"[4xCRSR-RIGHT][32xSPACE]"
865  print"[4xCRSR-RIGHT][SPACE][PIJL L
     INKS]-toets[SPACE]=[SPACE]escape[S
     PACE]op[SPACE]elk[SPACE]moment[SPA
     CE]"
870  print"[4xCRSR-RIGHT][11xSPACE]beha
     lve[SPACE]by[SPACE]enter[SPACE]en[
     2xSPACE]"
875  print"[4xCRSR-RIGHT][11xSPACE]zoek
     en[15xSPACE]"
880  print"[4xCRSR-RIGHT][32xSPACE]"
885  print"[4xCRSR-RIGHT][32xSPACE]"
890  print"[4xCRSR-RIGHT][32xSPACE][HOM
     E][2xCRSR-DOWN][8xCRSR-RIGHT]";
895  gosub480:goto45
900  ifa$="n"then625
905  ifa$="j"thenprint"[SHIFT-CLR]":got
     o45
910  goto855
915  return
920  print"[HOME][11xCRSR-DOWN][12xCRSR
     -RIGHT][CTRL-9][15xSPACE][CTRL-0]"
925  print"[12xCRSR-RIGHT][CTRL-9][SPAC
     E]lege[2xSPACE]sheet[SPACE]![SPACE
     ][CTRL-0]"
930  print"[12xCRSR-RIGHT][CTRL-9][15xS
     PACE][CTRL-0]"
935  print"[12xCRSR-RIGHT][CTRL-9][SPAC
     E]zeker[SPACE]weten[SPACE]![SPACE]
     [CTRL-0]"
940  print"[12xCRSR-RIGHT][CTRL-9][4xSP
     ACE](j/n)[6xSPACE][CTRL-0]"
945  print"[12xCRSR-RIGHT][CTRL-9][15xS
     PACE][CTRL-0]"
```

```
950   geta$:ifa$=""then950
955   ifa$="n"then45
960   ifa$="j"thenrun
965   goto950
970   print"[HOME][2xCRSR-DOWN]zoek[SPAC
      E]gegeven[SPACE]:[CTRL-4]";:
975   poke19,1:inputzk$:poke19,0
980   bl=len(zk$)
985   fori=1tocd:print"[HOME][3xCRSR-DOW
      N]";
990   ifzk$=left$(cm$(i),bl)then1005
995   next
1000  print"[CRSR-UP]komt[SPACE]niet[SPA
      CE]in[SPACE]het[SPACE]bestand[SPAC
      E]voor!":gosub710:goto50
1005  ifi>100thenpc=int(i/100)+1:goto1015
1010  pc=1
1015  a=int(i-((pc-1)*100))
1020  print"[CRSR-UP]gegeven[SPACE]staat
      [SPACE]in[SPACE]cel[SPACE]"chr$(pc
      +64);a
1025  gosub710:goto50
```

EINDE LISTING spreadsheet 64

Checksum spreadsheet 64

REGEL 5	36	REGEL 270	36	REGEL 535	246	REGEL 800	174
REGEL 10	0	REGEL 275	150	REGEL 540	4	REGEL 805	169
REGEL 15	201	REGEL 280	56	REGEL 545	234	REGEL 810	41
REGEL 20	247	REGEL 285	129	REGEL 550	166	REGEL 815	52
REGEL 25	172	REGEL 290	82	REGEL 555	24	REGEL 820	81
REGEL 30	187	REGEL 295	238	REGEL 560	24	REGEL 825	182
REGEL 35	115	REGEL 300	171	REGEL 565	47	REGEL 830	81
REGEL 40	247	REGEL 305	64	REGEL 570	186	REGEL 835	196
REGEL 45	96	REGEL 310	68	REGEL 575	252	REGEL 840	134
REGEL 50	141	REGEL 315	1	REGEL 580	225	REGEL 845	120
REGEL 55	187	REGEL 320	146	REGEL 585	198	REGEL 850	46
REGEL 60	36	REGEL 325	41	REGEL 590	29	REGEL 855	156
REGEL 65	7	REGEL 330	79	REGEL 595	52	REGEL 860	81
REGEL 70	254	REGEL 335	81	REGEL 600	30	REGEL 865	165
REGEL 75	72	REGEL 340	78	REGEL 605	93	REGEL 870	244
REGEL 80	62	REGEL 345	83	REGEL 610	98	REGEL 875	29
REGEL 85	69	REGEL 350	130	REGEL 615	150	REGEL 880	81
REGEL 90	32	REGEL 355	112	REGEL 620	192	REGEL 885	81
REGEL 95	116	REGEL 360	230	REGEL 625	13	REGEL 890	169
REGEL 100	102	REGEL 365	194	REGEL 630	87	REGEL 895	85
REGEL 105	143	REGEL 370	57	REGEL 635	105	REGEL 900	120
REGEL 110	125	REGEL 375	35	REGEL 640	99	REGEL 905	115
REGEL 115	41	REGEL 380	94	REGEL 645	187	REGEL 910	43
REGEL 120	118	REGEL 385	141	REGEL 650	254	REGEL 915	142
REGEL 125	30	REGEL 390	47	REGEL 655	130	REGEL 920	171
REGEL 130	62	REGEL 395	17	REGEL 660	168	REGEL 925	148
REGEL 135	122	REGEL 400	148	REGEL 665	251	REGEL 930	221
REGEL 140	103	REGEL 405	249	REGEL 670	27	REGEL 935	2
REGEL 145	108	REGEL 410	129	REGEL 675	92	REGEL 940	245
REGEL 150	113	REGEL 415	123	REGEL 680	125	REGEL 945	221
REGEL 155	129	REGEL 420	211	REGEL 685	99	REGEL 950	107
REGEL 160	114	REGEL 425	214	REGEL 690	195	REGEL 955	68
REGEL 165	135	REGEL 430	212	REGEL 695	207	REGEL 960	97
REGEL 170	28	REGEL 435	218	REGEL 700	130	REGEL 965	39
REGEL 175	103	REGEL 440	10	REGEL 705	37	REGEL 970	154
REGEL 180	121	REGEL 445	20	REGEL 710	165	REGEL 975	125
REGEL 185	190	REGEL 450	165	REGEL 715	130	REGEL 980	29
REGEL 190	32	REGEL 455	30	REGEL 720	52	REGEL 985	112
REGEL 195	178	REGEL 460	99	REGEL 725	81	REGEL 990	148
REGEL 200	168	REGEL 465	76	REGEL 730	174	REGEL 995	130
REGEL 205	41	REGEL 470	42	REGEL 735	103	REGEL 1000	64
REGEL 210	92	REGEL 475	17	REGEL 740	129	REGEL 1005	244
REGEL 215	177	REGEL 480	99	REGEL 745	64	REGEL 1010	118
REGEL 220	149	REGEL 485	80	REGEL 750	112	REGEL 1015	59
REGEL 225	246	REGEL 490	95	REGEL 755	172	REGEL 1020	146
REGEL 230	248	REGEL 495	101	REGEL 760	187	REGEL 1025	76
REGEL 235	142	REGEL 500	85	REGEL 765	172		
REGEL 240	57	REGEL 505	142	REGEL 770	200		
REGEL 245	41	REGEL 510	87	REGEL 775	33		
REGEL 250	99	REGEL 515	26	REGEL 780	81		
REGEL 255	231	REGEL 520	145	REGEL 785	81		
REGEL 260	125	REGEL 525	185	REGEL 790	61		
REGEL 265	57	REGEL 530	74	REGEL 795	83		

PRINT OUT C-128 met o.a. Briefmaker

Super Change 128

Van de zelfde maker als het vorige programma, Christiaan Duijker, is Super Change 128. Na het runnen van dit programma is het op een eenvoudige manier mogelijk verschillende toetsen een bepaalde funktie te geven. Dit kan zeker bij het schrijven van programmatuur veel tijd kunnen besparen.

```
1      rem
2      rem *** powersoft inc 1989 ***
3      rem ***                    ***
4      rem *** Christiaan Duijker ***
5      rem ***                    ***
6      rem ***     Zoetermeer     ***
7      rem
10     gosub 530
20     scnclr
30     gosub500
40     print:print"[SPACE][CTRL-9][SPACE]
       >>[SPACE]1.[SPACE]verander[SPACE]h
       elp[SPACE]toets.[5xSPACE]<<"
50     print"[SPACE][CTRL-9][SPACE]>>[SPA
       CE]2.[SPACE]verander[SPACE]shift[S
       PACE]run/stop[2xSPACE]<<"
60     print"[SPACE][CTRL-9][SPACE]>>[SPA
       CE]3.[SPACE]verander[SPACE]functie
       [SPACE]toetsen[SPACE]<<"
70     print:print"[SPACE][CTRL-9][SPACE]
       >>[SPACE]maak[SPACE]uw[SPACE]keuze
       [SPACE]<[SPACE]1-3[SPACE]>[7xSPACE
       ]<<[CTRL-0]"
80     getkeykz$:k=val(kz$):ifk<1ork>3the
       ngoto80
90     on k gosub 110,240,370
100    scnclr:goto30
110    scnclr
120    gosub500
130    print:print"[SPACE][CTRL-9][SPACE]
       >>[SPACE]geef[SPACE]nieuw[SPACE]he
       lp[SPACE]comm.[7xSPACE]<<[SPACE][C
       TRL 0]"
140    j=1:inputn$
150    print"[SPACE][CTRL-9][SPACE]>>[SPA
       CE]met[SPACE]auto-return[SPACE](j/
       n)[9xSPACE]<<[SPACE][CTRL-0]"
160    getkeya$
170    ifa$="j"thenn$=n$+chr$(13)
180    poke 4105,len(n$)
190    fori=4096to4104:a=a+peek(i):nexti
200    fori=4106+a to 4106+a+len(n$):n=as
       c(mid$(n$,j,1))
210    if n=64thenpoke i,13:j=j+1:nexti
220    poke i,n:j=j+1:nexti
230    return
240    scnclr
250    gosub500
260    print:print"[SPACE][CTRL-9][SPACE]
       >>[SPACE]geef[SPACE]nieuw[SPACE]sh
       ift[SPACE]run/stop[SPACE]comm.<<[S
       PACE][CTRL-0]"
270    j=1:inputn$
280    print"[SPACE][CTRL-9][SPACE]>>[SPA
       CE]met[SPACE]auto-return[SPACE](j/
       n)[10xSPACE]<<[SPACE][CTRL-0]"
290    getkeya$
300    ifa$="j"thenn$=n$+chr$(13)
310    poke 4104,len(n$)
320    fori=4096to4103:a=a+peek(i):nexti
330    fori=4106+a to 4106+a+len(n$):n=as
       c(mid$(n$,j,1))
340    if n=64thenpoke i,13:j=j+1:nexti
350    poke i,n:j=j+1:nexti
360    return
370    scnclr
380    gosub500
390    print:print"[SPACE][CTRL-9][SPACE]
       >>[SPACE]welke[SPACE]functie[SPACE
       ]toets[SPACE]?[11xSPACE]<<[SPACE][
       CTRL-0]"
400    getkeya$:a=val(a$):ifa<1ora>8theng
       oto400
410    print"[SPACE][CTRL-9][SPACE]>>[SPA
       CE]geef[SPACE]nieuw[SPACE]comm.[SP
       ACE]voor[SPACE]f.key[5xSPACE]<<[SP
       ACE][CTRL-0]"
420    input nf$
430    print"[SPACE][CTRL-9][SPACE]>>[SPA
       CE]met[SPACE]auto-return[SPACE](j/
       n)[11xSPACE]<<[SPACE][CTRL-0]"
440    getkeyar$
450    ifar$="j"thennf$=nf$+chr$(13):goto
       490
460    print"[SPACE][CTRL-9][SPACE]>>[SPA
       CE]met[SPACE]auto-aanhalingstekens
       [SPACE](j/n)[SPACE]<<[2xSPACE]"
470    getkeyar$
480    ifar$="y"thennf$=nf$+chr$(34)
490    key a,nf$:return
500    print"[CTRL-9]>>>[2xSPACE]super[SP
       ACE]key[SPACE]changer[2xSPACE](c)[
       SPACE]1989[5xSPACE]<<<[CTRL-0]"
510    return
520    print"[4xSPACE]"
530    scnclr
540    fori=1to5:reada$
550    print:printtab(20-len(a$)/2);a$
560    nexti:sleep5:return
570    data"powersoft[SPACE]presents:"
580    data"super[SPACE]change[SPACE]128"
590    data"(c)[SPACE]1989[SPACE]by[SPACE
       ]c.duijker"
600    data"voor[SPACE]commodore[SPACE]in
       fo"
610    data"alle[SPACE]rechten[SPACE]voor
       behouden"
```

EINDE LISTING super change 128

Checksum superchange 128

REGEL 1	143	REGEL 300	147
REGEL 2	9	REGEL 310	18
REGEL 3	127	REGEL 320	52
REGEL 4	178	REGEL 330	109
REGEL 5	249	REGEL 340	108
REGEL 6	213	REGEL 350	186
REGEL 7	143	REGEL 360	142
REGEL 10	37	REGEL 370	232
REGEL 20	232	REGEL 380	34
REGEL 30	34	REGEL 390	156
REGEL 40	82	REGEL 400	47
REGEL 50	130	REGEL 410	17
REGEL 60	203	REGEL 420	61
REGEL 70	157	REGEL 430	185
REGEL 80	228	REGEL 440	81
REGEL 90	131	REGEL 450	209
REGEL 100	14	REGEL 460	231
REGEL 110	232	REGEL 470	81
REGEL 120	34	REGEL 480	131
REGEL 130	106	REGEL 490	230
REGEL 140	94	REGEL 500	206
REGEL 150	185	REGEL 510	142
REGEL 160	255	REGEL 520	221
REGEL 170	147	REGEL 530	232
REGEL 180	19	REGEL 540	172
REGEL 190	53	REGEL 550	61
REGEL 200	109	REGEL 560	11
REGEL 210	108	REGEL 570	62
REGEL 220	186	REGEL 580	151
REGEL 230	142	REGEL 590	80
REGEL 240	232	REGEL 600	222
REGEL 250	34	REGEL 610	126
REGEL 260	41		
REGEL 270	94		
REGEL 280	185		
REGEL 290	255		

Briefmaker 128

Dit programma biedt U naast het menu gestuurd maken van een zakelijke brief, de mogelijkehid om komplete brieven op disk op te slaan. Standaard staan losse stukken tekst welke alle gekombineerd kunnen worden. Een verdere uitleg is eigenlijk niet mogelijk. U moet het zien en uitproberen om er achter te komen wat de mogelijkheden van dit programma zijn. Het werkt alleen met een 80 koloms monitor. Het programma is gemaakt door H.J. v Eelen.

```
1     rem *** briefmaker 128/80- h.j.c.
       van eeten ***
2     :
3     ifpeek(215)<>128thenprint"c-128/80
      [SPACE]kol.[SPACE]progamma":end
4     fast:r$=chr$(13):e$=chr$(27)+"u":y
      $=chr$(133):z$=chr$(138):trap5:got
      o7
5     trap5:gosub6:close4:printerr$(er)"
      [SPACE]druk[SPACE]een[SPACE]toets"
      ;:getkeya$:goto16
6     printchr$(19)chr$(19):scnclr:retur
      n
7     printchr$(142):color5,7:gosub6:cha
      r1,25,6,"maak[SPACE]uw[SPACE]zakel
      ijke[SPACE]brieven[SPACE]met"
8     char1,31,9,"[SPACE]briefmaker[SPAC
      E]128[SPACE]",1:char1,29,20,"door[
      SPACE]h.j.c.[SPACE]van[SPACE]eeten
      ":sleep4
9     b=0:fork=5141to5394:reada:pokek,a:
      b=b+a:next
10    ifb<>29126thenprintr$"fout[SPACE]i
      n[SPACE]dataregels":end
11    dimtr$(25,25):ar=0:pl=72:fork=1to2
      5:l$=l$+chr$(125):next
12    fork=1to60:c$=c$+chr$(32):next:gos
      ub6:ifpeek(254)<>9thensys5347:poke
      254,9
13    key1,y$:key2,chr$(137):key4,z$:key
      7,chr$(136)
14    :
15    rem                    menu
16    printchr$(142):color5,8:gosub6:cha
      r1,27,3,"[SPACE]briefmaker[SPACE]h
      oofdmenu[SPACE]",1
17    window20,5,60,24:printr$"1.[SPACE]
      brief[SPACE]laden[SPACE]van[SPACE]
      disk"
18    printr$"2.[SPACE]brief[SPACE]maken
      ":ifpl<12thenpl=72
19    printr$"3.[SPACE]brief[SPACE]wegsc
      hrijven[SPACE]naar[SPACE]disk"
20    printr$"4.[SPACE]pagina[SPACE]leng
      te[SPACE]instellen[SPACE]([SPACE]i
      s[SPACE]"pl")"
21    printr$"5.[SPACE]print[SPACE]brief
      [SPACE]op[SPACE]beeldscherm[SPACE]
      of[SPACE]printer"e$r$
22    poke208,0:getkeya$:gosub6:onval(a$
      )+1goto16,25,43,26,129,135
23    :
24    rem                    brief laden/we
      gschrijven
25    c=1:b$="r":color5,11:char1,9,6,"[S
      PACE]brief[SPACE]laden[SPACE]",1:g
      oto27
26    c=2:b$="w":color5,4:char1,9,6,"[SP
      ACE]brief[SPACE]wegschrijven[SPACE
      ]",1
27    window0,9,50,24,1:poke208,0:input"
      progammanaam[SPACE](d=dir,[SPACE]m
      =menu)";n$
28    n$=left$(n$,14):ifn$="d"thenwindow
      52,0,79,24,1:directory:goto27
29    ifn$="m"then16
30    printr$spc(33)"8"chr$(145):input"d
      evice[SPACE]nr.[SPACE](disk=8[SPAC
      E]datasette=1)";d
31    printr$"disk[SPACE]of[SPACE]casset
      te[SPACE]in[SPACE]orde[SPACE](j/n)
      ?":poke208,0:getkeya$:ifa$<>"j"the
      n31
32    open4,(d),4,(n$)+",s,"+b$:ongosub
      34,35:fork=0to24:forl=0to24:ongos
      ub36,39
33    nextl,k:close4:printr$r$ds$spc(3)"
      druk[SPACE]een[SPACE]toets":poke20
      8,0:getkeya$:goto16
34    input#4,h$:pl=val(h$):return
35    h$=str$(pl):print#4,h$:return
36    input#4,h$:a$="":a=len(h$)-1:ifa<1
      thentr$(k,l)="":return
37    form=1toa:g$=chr$(abs(asc(mid$(h$,
      m,1))-255)):a$=a$+g$:next
38    tr$(k,l)=a$:return
39    h$=tr$(k,l)+b$:a=len(h$):a$="":for
      m=1toa:g$=chr$(abs(asc(mid$(h$,m,1
      ))-255))
40    a$=a$+g$:next:print#4,(a$):return
41    :
42    rem                         brief maken
```

```
43   color5,6:char1,31,6,"[SPACE]brief[
     SPACE]maken[SPACE]",1:window28,9,7
     9,24
44   printchr$(14)"1.[SPACE]VOLLEDIGE[S
     PACE]BRIEF"r$r$"2.[SPACE]INVOEREN[
     SPACE]PER[SPACE]ALINEA"r$r$"3.[SPA
     CE]MENU"
45   poke208,0:getkeym$:ifm$="2"then71
46   ifm$<>"1"then16
47   gosub6:char1,34,0,"[SPACE]BRIEFHOO
     FD[SPACE]",1:window15,3,79,24
48   print"VOORBEELD[SPACE]AFZENDER:"r$
     r$"B.F.[SPACE]De[SPACE]Vries"r$"Bo
     slaan[SPACE]65"
49   print"1200[SPACE]BA[2xSPACE]Bosdam
     "r$"telefoon[SPACE]017-737747"r$
50   print"GEEF[SPACE]NU[SPACE]IN[SPACE
     ]MAXIMAAL[SPACE]8[SPACE]REGELS[SPA
     CE]UW[SPACE]AFZENDERGEGEVENS"
51   print"(DRUK[SPACE]RETURN[SPACE]ALS
     [SPACE]U[SPACE]NIETS[SPACE]MEER[SP
     ACE]WILT[SPACE]INVULLEN)":poke208,0
52   fork=1to8:inputaf$(k):next:sleep1:
     printr$"DOE[SPACE]HETZELFDE[SPACE]
     MET[SPACE]DE[SPACE]ADRESSERING"
53   fork=1to8:poke208,0:inputad$(k):ne
     xt:sleep1:printr$"VOORBEELDEN[SPAC
     E]ONDERWERP:"
54   print"aangifte[SPACE]schadegeval"r
     $"sollicitatie[SPACE]vacature[SPAC
     E]agrarisch[SPACE]medewerker"
55   print"uw[SPACE]brief[SPACE]BVD/007
     [SPACE]7[SPACE]van[SPACE]8-8-1988"
     r$:poke208,0:input"betreft";bt$
56   print:poke208,0:input"de[SPACE]pla
     ats[SPACE]van[SPACE]waaruit[SPACE]
     u[SPACE]schrijft";pl$
57   print:poke208,0:input"datum[SPACE]
     (b.v.:[SPACE]31[SPACE]september[SP
     ACE]1988)";da$
58   printr$"KIES[SPACE]EEN[SPACE]AANHE
     F":a$(1)="Mevrouw,[SPACE]meneer,":
     a$(2)="Mijne[SPACE]heren,"
59   a$(3)="Geachte[SPACE]heer,":a$(4)=
     "Zeer[SPACE]geachte[SPACE]heer,":p
     oke208,0
60   print"1.[SPACE]"a$(1)r$"2.[SPACE]"
     a$(2)r$"3.[SPACE]"a$(3)r$"4.[SPACE
     ]"a$(4)r$"5.[SPACE]Geacht[SPACE]..
     .,"
61   print"6.[SPACE]Geachte[SPACE]...,"
     r$"7.[SPACE]Beste[SPACE]...,"r$"8.
     [SPACE]...,"r$:getkeya$:a=val(a$)
62   a$(5)="Geacht[SPACE]":a$(6)="Geach
     te[SPACE]":a$(7)="Beste[SPACE]":if
     a>0anda<5thenah$=a$(a)
63   r=0:ifa>4anda<9thenpoke208,0:input
     "...";a$:ah$=a$(a)+a$+","
64   fork=0to25:tr$(0,k)="":next:fork=1
     to8:ifaf$(k)<>""thentr$(0,r)=af$(k
     ):r=r+1
65   next:r=r+2:fork=1to8:ifad$(k)<>""t
     hentr$(0,r)=ad$(k):r=r+1
66   next:r=r+1:tr$(0,r)="betreft:[SPAC
     E]"+bt$:r=r+3:h$=pl$+",[SPACE]"+da
     $:h=60-len(h$)
67   tr$(0,r)=left$(c$,h)+h$:r=r+2:fork
     =0tor:iftr$(0,k)=""thentr$(0,k)=c$
68   next:tr$(0,r)=ah$:tr$(0,r+1)=c$
69   :
70   rem invoeren tekst per alinea
71   gosub6:char1,26,0,"[SPACE]INVOEREN
     [SPACE]TEKST[SPACE]PER[SPACE]ALINE
     A[SPACE]",1:al=1:window6,4,79,24
72   print"Denk[SPACE]bij[SPACE]het[SPA
     CE]invoeren[SPACE]van[SPACE]tekst[
     SPACE]aan[SPACE]de[SPACE]OPBOUW[SP
     ACE]van[SPACE]uw[SPACE]brief:"
73   printr$"Alinea[SPACE]0[SPACE]is[SP
     ACE]gereserveerd[SPACE]voor[SPACE]
     het[SPACE]hoofd[SPACE]van[SPACE]de
     [SPACE]brief."
74   printr$"Vanaf[SPACE]alinea[SPACE]1
     [SPACE]kunt[SPACE]u[SPACE]uw[SPACE
     ]eigen[SPACE]alinea's[SPACE]invoer
     en."
75   printr$"ALINEA[SPACE]1[SPACE]moet[
     SPACE]KORT[SPACE]duidelijk[SPACE]m
     aken[SPACE]waar[SPACE]de[SPACE]bri
     ef[SPACE]over[SPACE]gaat."
76   print"B.v.:[SPACE]'Omdat[SPACE]ik[
     SPACE]mijn[SPACE]schade-aangiftefo
     rmulier[SPACE]kwijt[SPACE]ben,[SPA
     CE]meld[SPACE]ik[SPACE]u"
77   print"met[SPACE]deze[SPACE]brief,[
     SPACE]dat[SPACE]ik[SPACE]gisteren,
     [SPACE]dinsdag[SPACE]12[SPACE]juli
     [SPACE]1988,[SPACE]betrokken"
78   print"ben[SPACE]geweest[SPACE]bij[
     SPACE]een[SPACE]ongeval[SPACE]op[S
     PACE]de[SPACE]Velperweg[SPACE]in[S
     PACE]de[SPACE]gemeente[SPACE]Arnhe
     m.'"
79   printr$"In[SPACE]de[SPACE]MIDDELST
     E[SPACE]ALINEA'S[SPACE]geeft[SPACE
     ]u[SPACE]de[SPACE]DETAILS."
80   printr$"De[SPACE]LAATSTE[SPACE]ALI
     NEA[SPACE]moet[SPACE]KORT[SPACE]zi
     jn:[SPACE]b.v.:[SPACE]'[SPACE]Omda
     t[SPACE]ik[SPACE]mijn[SPACE]auto[S
     PACE]voor"
81   print"mijn[SPACE]beroep[SPACE]nauw
     elijks[SPACE]kan[SPACE]missen,[SPA
     CE]verzoek[SPACE]ik[SPACE]u[SPACE]
     de[SPACE]schade[SPACE]zo[SPACE]snel"
82   print"mogelijk[SPACE]te[SPACE]taxe
     ren.'"
83   printr$"Zie[SPACE]M.F.[SPACE]Steeh
     ouder[SPACE]e.a.,[SPACE]Leren[SPAC
     E]communiceren,[SPACE]ISBN[SPACE]9
     0[SPACE]01[SPACE]80816[SPACE]6."
84   printchr$(19)chr$(19):char1,31,24,
     "[SPACE]DRUK[SPACE]EEN[SPACE]TOETS
     [SPACE]",1:poke208,0:getkeya$
85   :
86   gosub6:window20,8,79,24:printr$spc
     (31);al;chr$(145)
87   poke208,0:input"ALINEA[SPACE](0-24
     [SPACE]OF[SPACE]EINDIG[SPACE]MET[S
     PACE]-1)";al
88   a$="ALINEA[SPACE]ERVOOR[SPACE]IS[S
     PACE]NOG[SPACE]LEEG":ifal<0then115
89   ifal>1theniftr$(al-1,0)=""thenal=a
     l-1:printa$:sleep4:goto86
90   gosub93:al=al+1:goto86
91   :
```

```
92    rem invoeren gegevens
93    gosub6:color5,3:window73,10,79,24
94    print"f2=[SPACE]verwijder"r$"aline
      a"r$r$"f4="r$"voeg"r$"alinea"r$"tu
      ssen"
95    window0,0,6,9:print"eindig"r$"met[
      SPACE]f1"r$r$"of"r$"zonder"r$"opsl
      ag"r$"met[SPACE]f7"
96    window73,0,79,3:print"alinea"al:po
      ke248,128:window9,0,9,24:printl$;
97    window70,0,70,24:printl$r$chr$(11)
      :window10,0,69,24:color5,6
98    fork=0to24:printtr$(al,k):next:pri
      ntchr$(19);
99    printe$;:getkeya$:ifa$<y$ora$>z$th
      enprinta$;:goto99
100   :
101   rem funktietoetsbewerkingen
102   a=asc(a$)-135:ifa>0thenonagoto108,
      109,111
103   h$=c$:bank1:p1=peek(pointer(h$)+1)
      :p2=peek(pointer(h$)+2):bank15
104   poke250,p1:poke251,p2:poke252,10:p
      oke253,0
105   fork=0to24:sys5141:tr$(al,k)=h$:next
106   c=0:fork=24to0step-1:iftr$(al,k)=c
      $andc=0thentr$(al,k)="":elsec=1
107   next
108   poke248,0:printchr$(12):gosub6:ret
      urn
109   fork=alto24:forl=0to25:tr$(k,l)=tr
      $(k+1,l):nextl,k
110   fork=0to25:tr$(25,k)="":next:poke2
      08,0:goto93
111   fork=24toalstep-1:forl=0to25:tr$(k
      +1,l)=tr$(k,l):nextl,k
112   fork=0to25:tr$(al,k)="":next:poke2
      08,0:goto93
113   :
114   rem ondertekening
115   ifm$="2"then16
116   gosub6:char1,33,0,"[SPACE]ONDERTEK
      ENING[SPACE]",1:window9,3,79,24:a$
      (1)="Hoogachtend,"
117   a$(2)="Met[SPACE]de[SPACE]meeste[S
      PACE]hoogachting,":a$(3)="Met[SPAC
      E]vriendelijke[SPACE]groet,"
118   print"KIES[SPACE]UIT:"r$"1.[SPACE]
      "a$(1)r$"2.[SPACE]"a$(2)r$"3.[SPAC
      E]"a$(3):poke208,0:getkeya$
119   a=val(a$):print:input"Uw[SPACE]naa
      m";nm$:input"evt.[SPACE]funktie[SP
      ACE](of[SPACE]druk[SPACE]RETURN)";f$
120   al=25:dowhiletr$(al-1,0)="":al=al-
      1:loop:iff$<>""thennm$=nm$+",[SPAC
      E]"+f$
121   tr$(al,0)=a$(a):fork=1to3:tr$(al,k
      )=c$:next:tr$(al,4)=nm$:a$=""
122   input"aantal[SPACE]bijlagen";a$:a=
      val(a$):ifa>0thenbegin:fork=5to7:t
      r$(al,k)=c$
123   next:tr$(al,8)="bijlagen:[SPACE]":
      ifa=1thentr$(al,8)="bijlage:[2xSPA
      CE]"
124   fork=8toa+7:input"bijlagenaam";b$:
      tr$(al,k)=tr$(al,k)+b$
125   tr$(al,k+1)=left$(c$,10):next:bend
126   goto16
127   :
128   rem                     pagina lengte
      instellen
129   color 5,5:char1,23,7,"[SPACE]pagin
      a[SPACE]lengte[SPACE]instellen[SPA
      CE]",1:window20,9,69,24
130   printr$"12[SPACE]inch[SPACE]papier
      [SPACE]telt[SPACE]meestal[SPACE]72
      [SPACE]regels"
131   printr$"11[SPACE]inch[SPACE]papier
      [SPACE]meestal[SPACE]66"r$r$"a4[SP
      ACE]telt[SPACE]70[SPACE]regels"
132   printr$spc(14);pl;chr$(145):input"
      pagina[SPACE]lengte";pl:goto16
133   :
134   rem                     print brief
135   printchr$(14):color5,15:char1,33,9
      ,"[SPACE]PRINT[SPACE]BRIEF[SPACE]"
      ,1:window9,12,79,24
136   print"PRINT[SPACE]OP[SPACE]SCHERM[
      SPACE]OF[SPACE]PRINTER[SPACE]OF[SP
      ACE]GA[SPACE]TERUG[SPACE]NAAR[SPAC
      E]MENU[SPACE](S/P/M)";
137   d=3:poke208,0:getkeys$:ifs$<>"s"an
      ds$<>"p"then16
138 ifs$="p"thend=4:printr$"PRINTER[SP
      ACE]OKE?[SPACE]DRUK[SPACE]EEN[SPAC
      E]TOETS";:getkeya$
139   gosub6:open4,d,7:al=0:r=1:dowhile(
      tr$(al,0)<>""andal<25)oral=0:ar=0:
      c=0
140   dowhiletr$(al,ar)<>""andar<25andtr
      $(al,ar)<>"":h$=tr$(al,ar):gosub14
      3:c=1
141   r=r+1:ar=ar+1:loop:ifc=1thenh$="":
      gosub143:r=r+1
142   al=al+1:loop:close4:print"DRUK[SPA
      CE]EEN[SPACE]TOETS";:getkeya$:gosu
      b6:goto135
143   a=r/(pl-9):ifa=int(a)thenfork=1to1
      0:print#4:next
144   ifd=3thenprintusing"####";r;:print
      spc(6);h$:return
145   print#4,spc(10);h$:return
146   :
147   rem  dataregels mac hinetaalroutines
148   data160,0,162,18,165,253,32,137,20
      ,232,165,252,32,137,20,162,31,32
149   data149,20,133,78,56,165,78,233,12
      8,48,2,133,78,56,165,78,133,79,233
      ,32
150   data48,7,56,165,78,233,96,48,7,24,
      165,78,105,64,133,78,56,165,79,233
      ,64
151   data48,14,56,165,79,233,95,16,7,24
      ,165,78,105,32,133,78,162,1,32,124
      ,20
152   data169,250,141,185,2,165,78,120,3
      2,119,255,88,200,192,60,208,161,16\2
153   data20,32,124,20,96,24,138,101,252
      ,133,252,165,253,105,0,133,253,96,
      142
154   data0,214,44,0,214,16,251,141,1,21
      4,96,142,0,214,44,0,214,16,251,173
      ,1
155   data214,96,162,16,134,252,162,18,3
      2,149,20,133,250,232,32,149,20,133
      ,251
```

```
156     data162,31,32,149,20,166,252,224,1
        4,48,2,24,74,224,11,16,2,24,10,133
        ,253
157     data134,252,165,250,162,18,32,137,
        20,165,251,232,32,137,20,162,31,16
        5,253
158     data32,137,20,166,252,202,208,193,
        96,120,162,18,169,32,32,137,20,232
        ,169
159     data0,32,137,20,160,255,192,81,240
        ,11,32,161,20,32,161,20,136,208,24
        3,88
160     data96,162,15,134,252,162,31,32,14
        9,20,166,252,202,208,244,240,235,4
        8
```

EINDE LISTING briefmaker 128

Checksum briefmaker 128

REGEL	1	119	REGEL 63	21	REGEL 125	198	
REGEL	2	58	REGEL 64	180	REGEL 126	240	
REGEL	3	125	REGEL 65	55	REGEL 127	58	
REGEL	4	144	REGEL 66	71	REGEL 128	172	
REGEL	5	36	REGEL 67	206	REGEL 129	236	
REGEL	6	135	REGEL 68	219	REGEL 130	8	
REGEL	7	255	REGEL 69	58	REGEL 131	22	
REGEL	8	134	REGEL 70	17	REGEL 132	122	
REGEL	9	194	REGEL 71	13	REGEL 133	58	
REGEL	10	154	REGEL 72	115	REGEL 134	132	
REGEL	11	17	REGEL 73	103	REGEL 135	74	
REGEL	12	76	REGEL 74	125	REGEL 136	89	
REGEL	13	112	REGEL 75	71	REGEL 137	158	
REGEL	14	58	REGEL 76	199	REGEL 138	249	
REGEL	15	196	REGEL 77	181	REGEL 139	93	
REGEL	16	136	REGEL 78	210	REGEL 140	143	
REGEL	17	199	REGEL 79	175	REGEL 141	2	
REGEL	18	92	REGEL 80	9	REGEL 142	92	
REGEL	19	248	REGEL 81	181	REGEL 143	223	
REGEL	20	159	REGEL 82	52	REGEL 144	206	
REGEL	21	0	REGEL 83	56	REGEL 145	151	
REGEL	22	27	REGEL 84	116	REGEL 146	58	
REGEL	23	58	REGEL 85	58	REGEL 147	251	
REGEL	24	25	REGEL 86	58	REGEL 148	96	
REGEL	25	201	REGEL 87	31	REGEL 149	148	
REGEL	26	160	REGEL 88	114	REGEL 150	166	
REGEL	27	177	REGEL 89	49	REGEL 151	121	
REGEL	28	18	REGEL 90	11	REGEL 152	21	
REGEL	29	78	REGEL 91	58	REGEL 153	90	
REGEL	30	112	REGEL 92	73	REGEL 154	13	
REGEL	31	174	REGEL 93	239	REGEL 155	146	
REGEL	32	163	REGEL 94	122	REGEL 156	130	
REGEL	33	218	REGEL 95	55	REGEL 157	195	
REGEL	34	34	REGEL 96	185	REGEL 158	160	
REGEL	35	53	REGEL 97	39	REGEL 159	132	
REGEL	36	155	REGEL 98	246	REGEL 160	10	
REGEL	37	78	REGEL 99	41			
REGEL	38	189	REGEL 100	58			
REGEL	39	79	REGEL 101	101			
REGEL	40	253	REGEL 102	186			
REGEL	41	58	REGEL 103	114			
REGEL	42	99	REGEL 104	176			
REGEL	43	171	REGEL 105	142			
REGEL	44	137	REGEL 106	94			
REGEL	45	63	REGEL 107	130			
REGEL	46	227	REGEL 108	164			
REGEL	47	231	REGEL 109	11			
REGEL	48	83	REGEL 110	148			
REGEL	49	237	REGEL 111	144			
REGEL	50	193	REGEL 112	186			
REGEL	51	25	REGEL 113	58			
REGEL	52	103	REGEL 114	92			
REGEL	53	56	REGEL 115	50			
REGEL	54	77	REGEL 116	21			
REGEL	55	20	REGEL 117	143			
REGEL	56	96	REGEL 118	187			
REGEL	57	38	REGEL 119	94			
REGEL	58	3	REGEL 120	130			
REGEL	59	120	REGEL 121	227			
REGEL	60	46	REGEL 122	220			
REGEL	61	93	REGEL 123	238			
REGEL	62	102	REGEL 124	101			

Checksum c16

```
10      rem ******************************
20      rem syntax.checksum
30      rem voor c-16 & plus/4
40      rem
50      rem syntax testen met 'sys 1536'
60      rem
70      rem v.851128.16       jan bodzinga
80      rem ******************************
90      i=1536        :rem beginadres
100     reada:ifa>=0then pokei,a:i=i+1:got
        o100
110     print"data[SPACE]is[SPACE]weggezet"
120     print"cheksum[SPACE]printen[SPACE]
        met[SPACE]'sys[SPACE]1536'
130     end
200     data 165, 43,166, 44,133
210     data  31,134, 32,169,147
220     data  32,210,255,160,  0
230     data 240,  3, 32, 73,  6
240     data  32, 73,  6,208,  1
250     data  96, 72,152, 32,131
260     data   6,168,104,234, 32
270     data  81,  6, 32, 73,  6
280     data 240, 12,201, 32,240
290     data 247, 24,101,252,133
300     data 252, 76, 37,  6,166
310     data 252,169,  0,132,253
320     data  32, 95,164,169, 13
330     data  32,210,255,164,253
340     data  76, 17,  6,200,208
350     data   2,230, 32,177, 31
360     data  96,162,  0,189,123
370     data   6,240,  6, 32,210
380     data 255,232,208,245, 32
390     data  73,  6,170, 32, 73
400     data   6,132,253, 32, 95
410     data 164,162,  3,169, 32
420     data  32,210,255,202,208
430     data 250,169,  0,133,252
440     data 164,253, 96, 82, 69
450     data  71, 69, 76, 32,  0
460     data   0, 72,138, 72, 32
470     data 225,255,240,251,104
480     data 170,104, 96, -1
```

** EINDE LISTING checks16

Checksum Checksum C-16

REGEL 10	249	REGEL 300	118
REGEL 20	247	REGEL 310	204
REGEL 30	121	REGEL 320	165
REGEL 40	143	REGEL 330	252
REGEL 50	75	REGEL 340	106
REGEL 60	143	REGEL 350	98
REGEL 70	8	REGEL 360	163
REGEL 80	249	REGEL 370	45
REGEL 90	103	REGEL 380	0
REGEL 100	2	REGEL 390	58
REGEL 110	245	REGEL 400	108
REGEL 120	237	REGEL 410	159
REGEL 130	128	REGEL 420	245
REGEL 200	210	REGEL 430	202
REGEL 210	208	REGEL 440	176
REGEL 220	142	REGEL 450	12
REGEL 230	1	REGEL 460	54
REGEL 240	3	REGEL 470	43
REGEL 250	157	REGEL 480	1
REGEL 260	155		
REGEL 270	215		
REGEL 280	186		
REGEL 290	248		

PRINT OUT C- 16 met Collis

Collis 16

Collis is weer een ouderwets aktiespel. Het is gemmakt door Peter Boersma uit Stadskanaal. De bedoeling van dit spel is om je tegenstander door middel van muren in te sluiten. Door het programma wordt ook de stand van zaken aangegeven. Door middel van het keuzemenu is het geheel nog wat moeilijker te maken.

```
1     rem -----------------------------
2     rem -     c o l l i s i o n    (ss
      s)-
3     rem -c-16------------------------
4     rem - voor : commodore info     -
5     rem - door : peter boersma  (mrt'8
      8)-
6     rem - lg-
7     rem -          enschede          -
8     rem - (c) social software services

9     rem -----------------------------
10    scnclr:vol8:gosub 1240:gosub 690
20    vol8:color 0,1:color1,2:color4,3,3
30    print"pb[SPACE]sss[SPACE]'88":scnc
      lr
40    if q=1 then goto 120
50    rem ----- border ----------------
60    b=3112:e=3151:s=1:gosub 110
70    b=3151:e=4071:s=40:gosub 110
80    b=4071:e=4032:s=-1:gosub 110
90    b=4032:e=3112:s=-40:gosub 110
100   goto 120
110   fora=b to e step s:poke a,160:poke
       a-1024,87:next a:return
120   if u=1 then goto 180
130   rem ----- obstakels -------------
140   poke 3276,085:poke3277,073:poke331
      6,074:poke3317,075
150   poke 3824,085:poke3825,073:poke386
      4,074:poke3865,075
160   poke 3571,085:poke 3572,073:poke 3
      611,074:poke 3612,075
170   rem ----- initialisatie + score --
180   p1=3193:p2=3990:d1=1:d2=-1
190   print"[HOME][CTRL-3][5xSPACE][CTRL
       9][SPACE]speler[SPACE]1[SPACE]:";
      s1;tab(22)"[CTRL-6][SPACE]speler[S
      PACE]2[SPACE]:";s2;"[CTRL-2]"
200   rem ----- hoofdlus --------------
210   get a$
220   sound 3,1000,1
230   poke p1-1024,66:poke p2-1024,53
240   if a$="d" and d1<>1 then d1=-1
250   if a$="[CRSR-LEFT]" and d2<>1 then
       d2=-1
260   if a$="[CRSR-RIGHT]" and d2<>-1 th
      en d2=1
270   if a$="6" and d1<>-1 then d1=1
280   if a$="5" and d1<>40then d1=-40
290   if a$="[CRSR-UP]" and d2<>40then
      d2=-40
300   if a$="[CRSR-DOWN]" and d2<>-40th
      en d2=40
310   if a$="r" and d1<>-40then d1=40
320   p1=p1+d1:p2=p2+d2
330   if p1<3112 then p1=p1+960
340   if p2<3112 then p2=p2+960
350   if p1>4071 then p1=p1-960
360   if p2>4071 then p2=p2-960
370   if p1=p2 then w=0:goto 440
380   if peek(p1)<>32 then w=2:goto 440
390   if peek(p2)<>32 then w=1:goto 440
400   poke p1,081:poke p2,081
410   sound2,200,3:sound 1,700,1
420   for b=1 to 100-10*m:next b:goto 210
430   rem ----- uitslag op scherm ------
440   gosub 1260
450   on w+1 goto 460,540,610
460   scnclr
470   print"[4xCRSR-DOWN]"
480   color1,6,3:printtab(2)"UCIUCIUCIUC
      I[SPACE][COM-R][SPACE][COM-R].UCIU
      CIUCIUCI[COM-A]CI"
490   color1,8,5:printtab(2)"[SHIFT--][S
      PACE][COM-S][COM-Q]C[SPACE][COM-Q]
      C[SPACE][SHIFT--][SPACE][SHIFT--][
      SPACE][5xSHIFT--][SPACE][2xSHIFT--
      ][SPACE][SHIFT--][COM-Q]C[COM-W][C
      OM Q]C[COM-W][COM-Q][COM-R]K"
500   color1,3,4:printtab(2)"JCKJCKJCK[C
      OM E][SPACE][COM-E][SPACE]J[COM-E]
      K[2xCOM-E][SPACE][2xCOM-E][SPACE][
      2xCOM-E][SPACE][2xCOM-E][SPACE][2x
      COM-E]JC"
510   color1,8,5:print"[3xCRSR-DOWN][3xS
      PACE]druk[SPACE]op[SPACE]'return'[
      SPACE]om[SPACE]te[SPACE]starten":g
      osub 1500
520   geta$:if a$<>chr$(13) then 520
530   gosub 880:goto 20
540   s1=s1+1:scnclr:print"[4xCRSR-DOWN]"
550   color1,3,4:printtab(3)"[3xSHIFT--]
      .UCIUCIUCIUCI[COM-A]CI.[COM-A]CIUC
      IUCI[COM-A]CI"
560   printtab(3)"[5xSHIFT--][SPACE][2xS
      HIFT -][SPACE][SHIFT--][COM-Q]C[CO
      M W][COM-Q]C[COM-W][COM-Q][COM-R]K
      .[COM-Q][COM-R]K[SHIFT--][SPACE][2
      xSHIFT--][SPACE][2xSHIFT--][SPACE]
      [SHIFT--]"
570   printtab(3)"J[COM-E]K[2xCOM-E][SPA
      CE][2xCOM-E][SPACE][2xCOM-E][SPACE
      ][2xCOM-E][SPACE][2xCOM-E]JC[SPACE
      ][COM-E]JCJCKJCK[COM-Z]CK"
580   color1,8,5:print"[3xCRSR-DOWN][4xS
      PACE]druk[SPACE]op[SPACE]'return'[
      SPACE]om[SPACE]te[SPACE]starten":g
      osub 1500
590   geta$:if a$<>chr$(13) then 590
600   gosub 880:goto 20
610   s2=s2+1:scnclr:print"[4xCRSR-DOWN]"
620   color1,6,3:printtab(2)"[3xSHIFT--]
      .UCIUCIUCIUCI[COM-A]CI.UCI[COM-A]C
      IUCIUCIUCI"
630   printtab(2)"[5xSHIFT--][SPACE][2xS
      HIFT -][SPACE][SHIFT--][COM-Q]C[CO
      M W][COM-Q]C[COM-W][COM-Q][COM-R]K
      .[SHIFT--][SPACE][COM-S][COM-Q][CO
      M R]K[SHIFT--][SPACE][SHIFT--][COM
       Q]C[SPACE][SHIFT--][SPACE][SHIFT-
      -]"
640   printtab(2)"J[COM-E]K[2xCOM-E][SPA
      CE][2xCOM-E][SPACE][2xCOM-E][SPACE
      ][2xCOM-E][SPACE][2xCOM-E]JC[SPACE
```

```
     ]JCK[COM-E]JCJCKJCK[COM-E][SPACE][
     COM-E]"
650  color1,8,5:print"[3xCRSR-DOWN][3xS
     PACE]druk[SPACE]op[SPACE]'return'[
     SPACE]om[SPACE]verder[SPACE]te[SPA
     CE]gaan":gosub 1500
660  geta$:if a$<>chr$(13) then 660
670  gosub 880:goto 20
680  rem ----- intro ------------------
690  scnclr:color0,1:color1,2:color4,3,
     3
700  printchr$(14)chr$(8);
710  print"CoLLISIoN[21xSPACE](c)sss[SP
     ACE]'88"
720  print"[2xCRSR-DOWN][SPACE]Welkom[S
     PACE]bij[SPACE]'CoLLISIoN'[SPACE],
     [SPACE]een[SPACE]produkt"
730  print"[2xSPACE]van[SPACE]Social[SP
     ACE]Software[SPACE]Services[SPACE]
     (SSS)."
740  print"[CRSR-DOWN][SPACE]Bij[SPACE]
     dit[SPACE]spel[SPACE]voor[SPACE]tw
     ee[SPACE]spelers[SPACE]is[SPACE]he
     t"
750  print"[2xSPACE]de[SPACE]bedoeling[
     SPACE]de[SPACE]tegenstander[SPACE]
     in[SPACE]te"
760  print"[2xSPACE]sluiten[SPACE]zodat
     [SPACE]hij/zij[SPACE]tegen[SPACE]d
     e[SPACE]muur"
770  print"[2xSPACE]botst.[SPACE]Daarbi
     j[SPACE]maakt[SPACE]het[SPACE]niet
     [SPACE]uit[SPACE]of"
780  print"[2xSPACE]dat[SPACE]nou[SPACE
     ]jouw[SPACE]muur[SPACE]is[SPACE],[
     SPACE]of[SPACE]die[SPACE]van"
790  print"[2xSPACE]hem/haar[SPACE]zelf
     ."
800  print"[CRSR-DOWN][SPACE]De[SPACE]b
     esturing[SPACE]geschiedt[SPACE]als
     [SPACE]volgt:"
810  print"[CRSR-DOWN][2xSPACE]Speler[S
     PACE]1[SPACE](";:color1,3,4:print"
     ROOD";:color1,2:print")[SPACE]:joy
     stick[SPACE]1"
815 print"[5xSPACE]speler[SPACE]1[SPAC
     E]start[SPACE]LINKSBOVEN"
820  print"[2xSPACE]Speler[SPACE]2[SPAC
     E]([CTRL-6]GROEN[CTRL-2]):[SPACE]c
     ursor-toetsen"
825 print"[5xSPACE]speler[SPACE]2[SPAC
     E]start[SPACE]RECHTSONDER"
830  print"[CRSR-DOWN][2xSPACE](dit[SPA
     CE]is[SPACE]eventueel[SPACE]te[SPA
     CE]veranderen[SPACE]in[SPACE]de"
840  print"[2xSPACE]regels[SPACE]240-31
     0)"
850  print"[CRSR-DOWN][SPACE]Druk[SPACE
     ]op[SPACE]'RETURN'[SPACE]voor[SPAC
     E]het[SPACE]keuze-menu"
860  geta$:if a$<>chr$(13) then 860
870  rem ----- keuze - menu ----------
880  scnclr:if v=0then q=1:u=1:m=1
890  key1,"[COM-+]":key2,"[COM--]":key3
     ,"[COM-|]":key8,"[COM-*]"
900  color 0,1:color1,8,5:color 4,1
910  printchr$(142)
920  print"[SHIFT-CLR][2xCRSR-DOWN][5xS
     PACE][COM-R][SPACE][COM-R]UCI[COM-
     R][SPACE][COM-R][COM-A]C[COM-S]UCI
     [2xSPACE]U[COM-R]IUCIUCI[COM-R][SP
     ACE][COM-R]"
930  print"[5xSPACE][COM-Q][COM-R]K[COM
      Q]C[SPACE][SHIFT--][SPACE][SHIFT-
     -]UCK[COM-Q]C[SPACE]CC[3xSHIFT--][
     COM-Q]C[SPACE][SHIFT--][SPACE][2xS
     HIFT -][SPACE][SHIFT--]"
940  print"[5xSPACE][COM-E]JCJCKJCK[COM
      Z]C[COM-X]JCK[2xSPACE][COM-E][SPA
     CE][COM-E]JCK[COM-E][SPACE][COM-E]
     JCK":gosub 1500
950  if v=0then q$="zonder[SPACE]borde
     r"
960  if v=0then u$="zonder[SPACE]obsta
     kels":v=1
970  c$="snelheid[SPACE]:"+str$(m)+"[SP
     ACE]"
980  print"[HOME][7xCRSR-DOWN][SPACE]f1
     [SPACE]:[SPACE]";q$
990  print"[CRSR-DOWN][SPACE]f2[SPACE]:
     [SPACE]";u$
1000 print"[CRSR-DOWN][SPACE]f3[SPACE]:
     [SPACE]";c$
1010 print"[CRSR-DOWN][SPACE]f4[SPACE]:
     [SPACE]starten"
1020 get d$
1030 if d$="[COM-+]" and q=1 then q$="[
     CTRL-9][SPACE]met[SPACE]border[SPA
     CE][CTRL-0][2xSPACE]":q=0:goto 970
1040 if d$="[COM-+]" and q=0then q$="z
     onder[SPACE]border":q=1:goto 970
1050 if d$="[COM--]" and u=1 then u$="[
     CTRL-9][SPACE]met[SPACE]obstakels[
     SPACE][CTRL-0][2xSPACE]":u=0:goto
     970
1060 if d$="[COM--]" and u=0then u$="z
     onder[SPACE]obstakels":u=1:goto 970
1070 if d$="[COM-|]" then m=m+1:if m=11
      then m=1
1080 if d$="[COM-*]" then goto 1100
1090 goto 970
1100 print"[SHIFT-CLR][5xCRSR-DOWN]"
1110 color1,6,7:printtab(10)"[COM-A]CIU
     CIUCI[COM-A]CI[COM-R][SPACE][COM-R
     ]UCI"
1120 color1,6,6:printtab(10)"[SHIFT--][
     SPACE][2xSHIFT--][2xSPACE][SHIFT--
     ][SPACE][2xSHIFT--][SPACE][2xSHIFT
      -][SPACE][SHIFT--][2xSPACE][SHIFT
      -]"
1130 color1,6,5:printtab(10)"[COM-Q][CO
     M R]K[COM-Q]C[SPACE][COM-Q]C[COM-W
     ][SHIFT--][SPACE][SHIFT--]J[COM-R]
     K[SPACE]UK"
1140 color1,6,4:printtab(10)"[2xSHIFT--
     ][SPACE][SHIFT--][2xSPACE][SHIFT--
     ][SPACE][2xSHIFT--][SPACE][SHIFT--
     ][SPACE][SHIFT--][2xSPACE][SHIFT--
     ][SPACE]"
1150 color1,6,3:printtab(10)"[COM-E]JCJ
     CK[COM-E][SPACE][COM-E][COM-Z]CK[S
     PACE][COM-E][2xSPACE].[SPACE]"
1160 for t=1 to 1000:next t:print"[SHIF
     T CLR][5xCRSR-DOWN]"
1170 color1,6,7:printtab(10)"UCI[COM-A]
     [COM-R][COM-S]UCI[COM-A]CI[COM-A][
```

```
      COM-R][COM-S][SPACE][SHIFT--][SPAC
      E]"
1180  color1,6,6:printtab(10)"[SHIFT--][
      3xSPACE][SHIFT--][SPACE][SHIFT--][
      SPACE][2xSHIFT--][SPACE][SHIFT--][
      SPACE][SHIFT--][2xSPACE][SHIFT--][
      SPACE]"
1190  color1,6,5:printtab(10)"JCI[SPACE]
      [SHIFT--][SPACE][COM-Q]C[COM-W][CO
      M Q][COM-R]K[SPACE][SHIFT--][2xSPA
      CE][SHIFT--][SPACE]"
1200  color1,6,4:printtab(10)"[2xSPACE][
      SHIFT--][SPACE][SHIFT--][SPACE][SH
      IFT -][SPACE][3xSHIFT--][2xSPACE][
      SHIFT--][2xSPACE][SHIFT--][SPACE]"
1210  color1,6,3:printtab(10)"JCK[SPACE]
      [COM-E][SPACE][COM-E][SPACE][2xCOM
       E]JC[SPACE][COM-E][2xSPACE].[SPAC
      E]"
1220  for t=1 to 600:next t:return
1230  rem ----- titel-scherm -----------
1240  scnclr:color0,1:color1,6,3:color4,1
1250  printchr$(142)
1260  print"[SHIFT-CLR]"tab(16)"press[SP
      ACE]key":print"[3xCRSR-DOWN]"
1270  color1,6,3:printtab(10)"UCCCCC[SHI
      FT -]C[SHIFT--]CCCCCCCCC[COM-R]CI"
1280  color1,6,4:printtab(10)"[SHIFT--][
      5xSPACE][SHIFT--][SPACE][SHIFT--][
      SPACE].UCI.[3xSPACE][SHIFT--][SPAC
      E][SHIFT--]"
1290  color1,6,5:printtab(10)"[SHIFT--][
      2xSPACE]UCI[SHIFT--][SPACE][SHIFT-
      -][SPACE][SHIFT--]JCI[SHIFT--]UCI[
      SHIFT--][SPACE][SHIFT--]"
1300  color1,6,6:printtab(10)"[SHIFT--][
      2xSPACE][SHIFT--][SPACE][2xSHIFT--
      ][SPACE][SHIFT--][SPACE][SHIFT--][
      2xSPACE][3xSHIFT--][SPACE][2xSHIFT
       -][SPACE][SHIFT--]"
1310  color1,6,7:printtab(10)"JCKJCKJCJC
      JJCKJJCKK[SPACE]K"
1320  print"[CTRL-2]":for a=1 to 19:vol
      8-(20-a)/4
1330  poke 3753+a,81:poke 3752+a,32
1340  poke 3791-a,81:poke 3792-a,32
1350  sound 3,200-a*10,1:sound 1,200-a*1
      0-2,1
1360  next a
1370  p=3772:e=81:gosub 1420
1380  for e=5+2*16 to 5+7*16 step 16
1390  p=3772-1024:gosub 1420
1400  get a$:if a$<>"" then return
1410  next e:goto 1380
1420  for a=0to 7:vol 8-a
1430  sound 1,a*100+2+2*e,2:sound 2,a*10
      0+2*e,2
1440  poke p-2*a,e:poke p+2*a,e
1450  poke p+a*40,e:poke p-a*40,e
1460  poke p+a*41,e:poke p+a*39,e
1470  poke p-a*41,e:poke p-a*39,e
1480  next a:return
1490  rem ----- subr.'collision'--------
1500  print"[HOME][17xCRSR-DOWN]"
1510  printtab(13)"UI[2xSPACE]BB.UI.[2xS
      PACE]UI"
1520  printtab(13)"B[SPACE]UIBBBJIBUIBB"
1530  printtab(13)"JKJKJJJJKJJKJJ"
1540  return
1550  rem -pb--------------------sss'88-
```

EINDE LISTING collis 16

Checksum Collis 16

REGEL 1	92	REGEL 290	66	REGEL 660	83	REGEL 1010	195	REGEL 1380	121
REGEL 2	223	REGEL 300	194	REGEL 670	82	REGEL 1020	9	REGEL 1390	213
REGEL 3	127	REGEL 310	1	REGEL 680	38	REGEL 1030	156	REGEL 1400	13
REGEL 4	58	REGEL 320	227	REGEL 690	89	REGEL 1040	228	REGEL 1410	86
REGEL 5	187	REGEL 330	42	REGEL 700	161	REGEL 1050	200	REGEL 1420	184
REGEL 6	117	REGEL 340	45	REGEL 710	45	REGEL 1060	16	REGEL 1430	215
REGEL 7	127	REGEL 350	46	REGEL 720	202	REGEL 1070	92	REGEL 1440	125
REGEL 8	7	REGEL 360	49	REGEL 730	196	REGEL 1080	186	REGEL 1450	225
REGEL 9	92	REGEL 370	123	REGEL 740	110	REGEL 1090	41	REGEL 1460	233
REGEL 10	41	REGEL 380	37	REGEL 750	44	REGEL 1100	197	REGEL 1470	235
REGEL 20	132	REGEL 390	37	REGEL 760	191	REGEL 1110	32	REGEL 1480	139
REGEL 30	33	REGEL 400	245	REGEL 770	10	REGEL 1120	194	REGEL 1490	60
REGEL 40	130	REGEL 410	142	REGEL 780	74	REGEL 1130	119	REGEL 1500	17
REGEL 50	133	REGEL 420	52	REGEL 790	97	REGEL 1140	227	REGEL 1510	199
REGEL 60	127	REGEL 430	37	REGEL 800	224	REGEL 1150	47	REGEL 1520	42
REGEL 70	183	REGEL 440	86	REGEL 810	47	REGEL 1160	44	REGEL 1530	29
REGEL 80	46	REGEL 450	110	REGEL 815	98	REGEL 1170	28	REGEL 1540	142
REGEL 90	92	REGEL 460	232	REGEL 820	48	REGEL 1180	8	REGEL 1550	143
REGEL 100	28	REGEL 470	33	REGEL 825	41	REGEL 1190	85		
REGEL 110	84	REGEL 480	196	REGEL 830	227	REGEL 1200	6		
REGEL 120	140	REGEL 490	101	REGEL 840	31	REGEL 1210	165		
REGEL 130	187	REGEL 500	19	REGEL 850	126	REGEL 1220	202		
REGEL 140	104	REGEL 510	13	REGEL 860	85	REGEL 1230	208		
REGEL 150	118	REGEL 520	78	REGEL 870	114	REGEL 1240	91		
REGEL 160	96	REGEL 530	82	REGEL 880	156	REGEL 1250	72		
REGEL 170	103	REGEL 540	18	REGEL 890	41	REGEL 1260	167		
REGEL 180	22	REGEL 550	218	REGEL 900	61	REGEL 1270	87		
REGEL 190	91	REGEL 560	178	REGEL 910	72	REGEL 1280	44		
REGEL 200	164	REGEL 570	194	REGEL 920	179	REGEL 1290	66		
REGEL 210	6	REGEL 580	13	REGEL 930	253	REGEL 1300	124		
REGEL 220	87	REGEL 590	85	REGEL 940	81	REGEL 1310	197		
REGEL 230	123	REGEL 600	82	REGEL 950	101	REGEL 1320	71		
REGEL 240	141	REGEL 610	20	REGEL 960	198	REGEL 1330	7		
REGEL 250	232	REGEL 620	97	REGEL 970	221	REGEL 1340	15		
REGEL 260	104	REGEL 630	240	REGEL 980	200	REGEL 1350	87		
REGEL 270	127	REGEL 640	64	REGEL 990	84	REGEL 1360	195		
REGEL 280	228	REGEL 650	203	REGEL 1000	67	REGEL 1370	253		

GEOS info rubriek

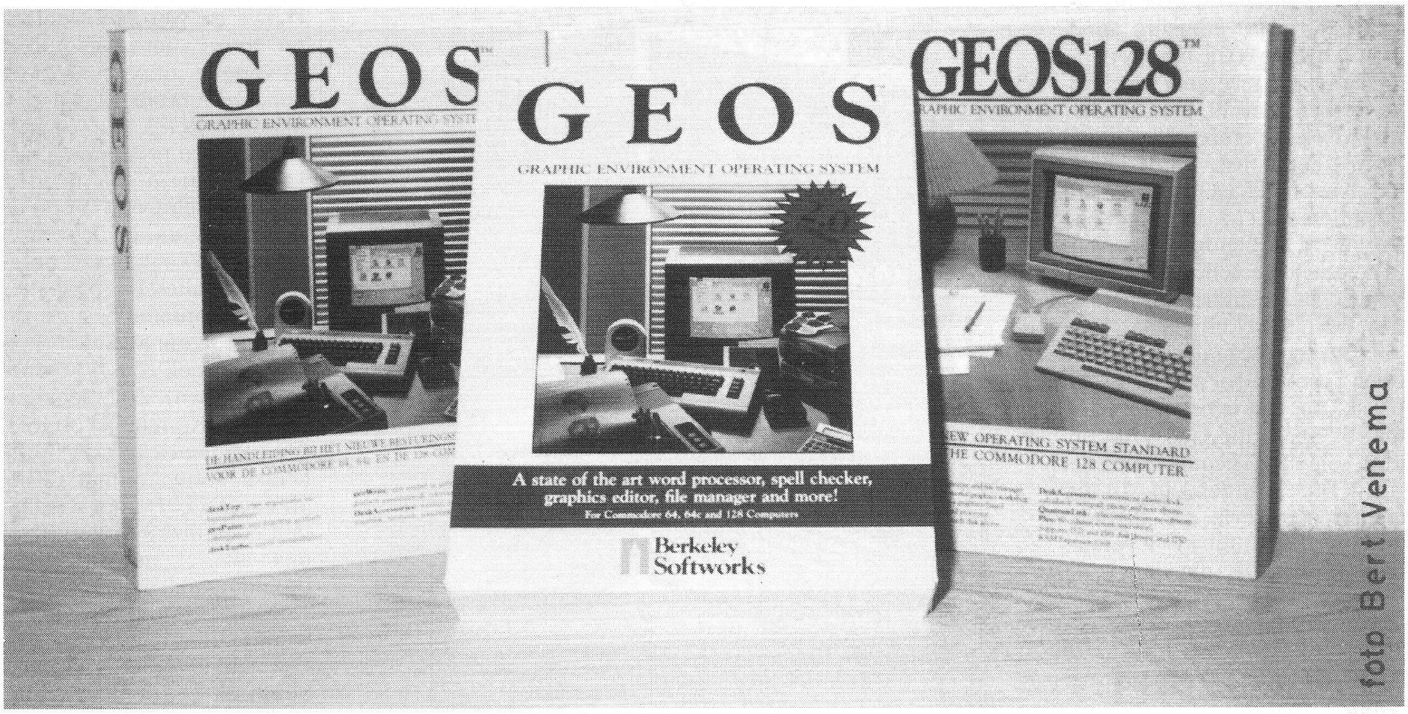


foto Bert Venema

De populaire informatie rubriek voor de aktieve GEOS gebruiker, voor al uw vragen, suggesties en ideeën.

Aanschaf diskdrive.

Dhr. Polman schrijft ons dat hij naast zijn huidige configuratie, bestaande uit een C64 en een cassetterecorder een diskdrive wil aanschaffen. Hem is verteld dat er meerdere diskdrives voor de C64 geschikt zijn. Dit is inderdaad het geval. Nu vraagt hij ons welke drive het meest geschikt is om met GEOS 2.0 te kunnen werken. De ervaring leert ons dat de Commodore diskdrives het meest geschikt zijn voor gebruik met GEOS 2.0, enwel het type 1541, als u met een C64 werkt en als u met een C128 gaat werken is een 1571 diskdrive ook zeer geschikt omdat u dan van de volle diskette capaciteit (360 Kb.) gebruik kunt maken. Andere merken diskdrives kunnen ook best goed werken met GEOS, maar daarmee hebben we tot op heden geen ervaring. Wat betreft de prijzen, de Commodore diskdrives zijn tegenwoordig niet zo prijzig meer als voor een paar jaar terug. Voor zo'n 300 gulden koopt u al een 1541 diskdrive. Voorts vraagt dhr. Polman ons wat de mogelijkheden van een REU (Ram Expansie Unit) zijn en welke modellen hierin verkrijgbaar zijn. Het zou te ver voeren om hier een hele verhandeling over een REU te gaan houden, maar in het kort komt het er op neer dat een REU (met GEOS) werkt als één soort diskdrive. Er zijn twee typen REU's in de handel, te weten de 1764 en 1751, respectievelijk voor de C64 en C128. De eerste wordt geconfigureerd als een 1541 drive met een opslagcapaciteit van 165 Kbyte en de tweede als een 1571 drive met een capaciteit van 331 Kbyte. Het enige verschil met een gewone diskdrive is dat u er geen diskettes in hoeft te plaatsen.

Aanschaf GEOS 2.0

Dhr. E. Vermeulen uit Vreeland overweegt de aanschaf van GEOS 2.0, maar heeft nog enkele vragen. Hij vindt dat de C64 van nature nogal traag is en een vrij kleine opslagcapaciteit heeft. Hieruit voortvloeiende luidt zijn vraag of dit met de aanschaf van GEOS 2.0 verholpen zal zijn. Ten eerste, de opslagcapaciteit zal door geen enkel programma worden vergroot, dus ook niet door GEOS. Wel wordt de opslagcapaciteit, voor wat betreft het interne geheugen, uitgebreidt als u een RAM expansie (zie hiervoor) aanschaft. Ten tweede wordt de snelheid wel enigszins verhoogd als u met GEOS 2.0 gaat werken. Door de aanwezigheid van de zgn. *diskTurbo* worden diskettemanipulaties zo'n 7 keer versneld. Verder zijn de applicaties, die onder het GEOS besturingssysteem allemaal snel, omdat ze zijn geschreven in een assembleertaal, hetgeen een extra vertaalslag overbodig maakt. Daarnaast is het gehele GEOS concept gebaseerd op een snelle werking, omdat gebruik gemaakt wordt van een scala aan standaard programma routines. Op uw vraag of u uw huidige programma's kunt blijven gebruiken, kunnen we zeggen dat dit inderdaad mogelijk is. Echter blijven deze programma's met hun eigen karakteristieken werken. GEOS wordt echter geleverd met standaard een tekstverwerker en een tekenprogramma. Voorts zijn er in de GEOS serie nog andere applicaties verkrijgbaar, zoals een spreadsheet, database, DeskTop Publishingpakket, een grafiekprogramma en nog vele andere utilities. Op uw laatste vraag, wat is een printerdriver. Dit is een stukje programma, wat u echter alleen nodig heeft bij gebruik van GEOS. Dit programma stuurt de betreffende printer aan, welke is aangesloten aan uw computer. De verschillende printerdrivers worden

standaard meegeleverd met het GEOS 2.0 besturingssysteem.

Installeren van GEOS.

Van de heer R. Sieben uit Stein ontvingen we een schrijven dat het installatieproces bij hem niet zo goed is verlopen. Nadat hij de *Systemdisk* had geïnstalleerd, wilde hij de *applications disk* installeren. Tot zijn verbazing stond hier niet GeoWrite op. Na contact met de leverancier gehad te hebben heeft hij het installatieproces stop gezet. Dit laatste is niet het slimste om te doen. Nimmer moet men tijdens het installeren het proces afbreken. Mijnheer Sieben refereert naar de zojuist verschenen Nederlandse basishandleiding, waarin een afbeelding staat met GeoWrite op één diskette tesamen met GeoPaint en de DeskTop. Het is best mogelijk dat de applicatie GeoWrite op de volgende pagina van de DeskTop is afgebeeld. Is dit echter niet geval dan dient u zich wederom tot uw leverancier te wenden, die dan voor een vervangend exemplaar dient zorg te dragen.

Vervangende GEOS disk.

Van Marco de Hoogh uit Alphen aan de Rijn ontvingen we een brief waarin hij schrijft dat hij vorig jaar het pakket GeoPublish had aangeschaft. Echter nadat deze was geinstalleerd met zijn GEOS diskette bleek dat zijn bootdiskette van GEOS niet juist werkte. Na contact gehad te hebben met de leverancier, krijgt hij een nieuwe GEOS bootdiskette toegestuurd. Echter nu blijkt zijn eerder gebruikte GeoPublish niet meer te werken. Dit klopt inderdaad, want als u een applicatie koopt en installeert met een GEOS bootdiskette, dan wordt op het moment van installeren een bepaalde code naar uw applicatiediskette geschreven.

Deze code voorkomt dat er illegale copieën van GEOS pakketten in de omloop komen. Hiermee is uw probleem dus niet opgelost. Er zijn nu twee mogelijkheden. De eerste is om contact op te nemen met de Stichting Geos Gebruikers in Almere, die telefonisch bereikbaar zijn onder nummer 03240 - 10041 of als u het schriftelijk wilt doen Postbus 52 1300 AB Almere. Deze Gebruikers Groep kan u wellicht helpen, omdat in deze gebruikers groep ook de support afdeling van de GEOS importeur is ondergebracht.
Een tweede mogelijkheid is de aanschaf van GEOS 2.0. Dit pakket is in staat om reeds eerder geinstalleerde applicatie in te sleutelen met de code die op één van uw applicaties is weggeschreven. Neem in dit geval contact op met uw leverancier en probeer het een en ander voor beide partijen gunstig op te lossen.



Stichting Geos Gebruikers.

We hebben al een paar keer eerder melding gemaakt van het bestaan van deze gebruikersgroep. Toch willen we nogmaals uw aandacht vestigen op deze stichting. Te meer omdat er voor een klein bedrag per jaar een hoop activiteiten voor u worden verwezenlijkt. Zo ontvangt u een aantal keren per jaar de GEOS Nieuwsbrief, met allerlei nuttige tips en trucs. Voorts wordt in dit periodiek aandacht besteed aan buitenlandse activiteiten op het gebied van GEOS. Naast dit alles worden er regelmatig bijeenkomsten georganiseerd, waar actieve GEOS gebruikers ervaringen en Public Domain software uitwisselen. Kortom bent u een actieve GEOS gebruiker, dan is deze gebruikers groep een must. Nog één keer het adres, Stichting Geos Gebruikers, Postbus 52, 1300 AB Almere.

Buitenlands GEOS nieuws.

Diegenen onder u die wel eens een buitenlands blad over Commodore computers aanschaft, zal inmiddels wel gemerkt hebben dat deze bladen ook ruime aandacht besteden aan de GEOS programmalijn. In Duitsland gaat men op dit gebied wel heel erg ver. Zo brengt het softwarehuis Markt & Technik de gehele GEOS-lijn in het Duits op de markt. Niet alleen de handleidingen zijn vertaald maar ook de programma's zelf zijn in het Duits geconverteerd. Helaas is de afzet van GEOS programma's in Nederland niet van zo'n omvangrijke schaal dat men het kostendekkend kan gaan vertalen. Maar wat die mensen in Duitsland doen heeft toch wel veel voordelen, zowel voor de gebruiker als voor de softwaremarkt op zich. De gebruiker voelt zich natuurlijk gelijk thuis in zo'n geavanceerd programma, dat dan ook nog in zijn moedertaal is geschreven. Voor de programmeurs is de omzet van de pakketten een stimulans om allerlei applicaties voor dit slinmme besturingssysteem te ontwikkelen (hetgeen hier ook eens zou moeten gebeuren). Zo bracht hetzelfde softwarehuis kortgeleden een nieuw pakket op de Duitse software markt, het *Mega Pack 1* voor GEOS. Een boek met 3 diskettes vol met allerlei nuttige utility programma's, die kunnen worden gebruikt in samenwerking met GEOS.
Een uitgebreide plaatjesbibliotheek, een converteerprogramma voor het omzetten van allerlei graphics, een verzameling letterfonts, een applicatie om een printerdriver aan te kunnen passen aan uw eigen configuratie en nog veel meer. Het bijbehorende boek telt maar liefst 120 pagina's. Het pakket wordt in Duitsland op de markt gebracht en is tevens in Nederland verkrijgbaar bij het Computer Collectief voor fl. 79,00.

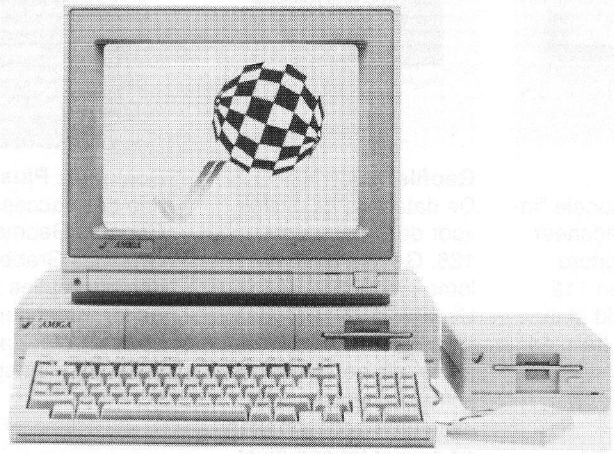
Foutje in Printshop Converter.

In de vorige Commodore Info werd aandacht besteed aan een drietal programma's van Patrick Cools uit Varsenaere. Hierbij was ondermeer het programma Printshop Converter. Helaas stonden hierin een tweetal kleine foutjes. Hieronder zijn de correcte regels weergegeven. De volgende regels moeten worden gewijzigd,
regel 370 moet worden *FOR I = 0 TO 5* en regel 410 moet worden *IF B = 33038 THEN 450*. Met deze correcties moet het geheel soepel verlopen.

Bert Venema

Binnenin AmigaDOS 9 (slot)

Net nu u gewend bent aan een vaste verschijning in uw favoriete blad, nooit gedacht dat ook in onze gelederen aan alles een einde komt. Nederland dreigt weer een verschijning armer te worden...dit, geachte lezers, zijn de laatste woorden die u ooit nog zult lezen van onze beruchte en toch ook tegelijkertijd befaamde cursus.



Zo we moesten even de zin afsluiten anders stond het zo lullig. Nee, wij gaan niet naar TV10, Veronique of Amiga magazine van Markt & Technik. Simpelweg, we zijn binnen de huidige doelstellingen van de cursus Binnenin AmigaDOS uitgesproken. Een aantal afleveringen geleden dreigden de donkere wolken van uitgesproken zinnen ons land der kennis al te overschaduwen. Echter bijtijds liet Commodore de zon van WorkBench 1.3 weer volop schijnen. Na een kleine spoedkursus WB1.3 en enkele dagen tot diep in het holst van de nacht experimenteren waren wij slagvaardig en konden onze vingers weer over het QWERTY toetsenbord laten flitsen. Helaas, dit keer dreigt het anders te gaan. We zien ons lot met gerechte rug en opgeheven hoofd tegemoet. Laat de golven der onwetendheid ons maar overspoelen, 'we shall overcome, i'll survive, we komen terug' of woorden van gelijke strekking. Het wachten is alleen op nieuwe ideeën, nieuwe vormgeving en WorkBench 1.4, want als we de geruchten mogen geloven zal dit ook niet al te lang meer duren.

Hoe nog 1 les vol te houden

De gedachte zou kunnen rijzen, 'when all words have been said, what's left to say?', of voor de rechtgeaarde Nederlander, als nu alles al behandeld is, wat moet er dan nog besproken worden? Een zeer juiste gedachte zegt ons filosofisch doch niet genoeg van logistiek doorgedrongen linkerhelft van de hersenen. We zouden drie bladzijden vol kunnen schrijven met hypothetisch geklets, dat zou echter leden kunnen kosten. We kunnen ook de Hilversumse methode handhaven, herhalen, herhalen en vooral 'repeaten' (Engels voor herhalen). Een 80 watts lichtje gaat ons op. Hadden we de vorige keer onzelf geen suggestie gedaan. Jawel. Even C-Info nummer 5 erop naslaan. Veel blablaa... eeuh...We gaan nog een keer in op het aspect "hoe eigen DOS.. zie je wel. We wisten het wel! Waren we weer eens wat vergeten! Hoeveel tijd tot de 'deadline'? 4 Dagen. Moet kunnen. Dit wordt 'stressen' en 'developpen'. Wat zou interresant kunnen zijn....eerst maar even stevig 'brainstormen'. Tot over een kwartiertje.

Eigen DOS commando's

Eigen DOS commando's. De droom van elke computerbezitter. Tenminste, van sommige dan.. U leest nog steeds? Goed, dan bent u dus één van de weinigen die uw eigen DOS commando's wilt programmeren. Het volgende vereist enige kennis van de taal C. (Niet zo heel veel, een tikkeltje)

Slechts weinige DOS commando's behoeven geen input van variabelen, dus er moet een mogelijkheid bestaan om variabelen vanuit de DOS omgeving naar het commando te transporteren. Hoe.., is de grote vraag. Het is niet zo moeilijk als het lijkt. Vrijwel elk C handboek geeft aan hoe dit te doen. Een C programma start op door de main() functie uit te voeren. Variabelen die aan het programma door moeten worden gegeven zullen dus op één of de andere manier in de main() functie ingelezen moeten worden. De oplossing luidt als volgt. Tussen de haakjes, '(' en ')', van de mainfunctie kunnen twee variabelen opgegeven worden, te weten *argv* en *argc*. Deze twee variabelen namen staan voor respectievelijk, argument variable en argument counter. In *argc*, de argument counter, wordt bijgehouden hoeveel variabelen van de commandoregel worden 'doorgesluist' naar het programma. In de array *argv[]*, de argumentvariabele(s), worden de pointers naar de variabelen vermeldt. Door nu *argv[]* als pointer naar een char of UBYTE te declareren kunt u de strings (lees variabelen) rechtstreeks uitlezen, afprinten, vergelijken of wat dan meer. That's all! Meer is er niet aan. Doodsimpel nietwaar! Het wordt nu eerst tijd voor een voorbeeld. Bekijk programma 1. Het doet niets meer dan nagaan hoeveel variabelen in worden gegeven en het beeldt deze vervolgens in het DOS venster af.

Nieuwe DOS commando's

Als klapstuk hebben we drie nieuwe commando's geprogrammeerd. De commando's heten, *Rect, Line* en *Circle* en u kunt ze vinden in dit artikel als programma 2,3 en 4. Zoals de namen al doen vermoeden zijn ze bedoeld voor grafisch tekenwerk in het DOS venster. Van elk van de commando's volgt nu een korte beschrijving zoals u van ons gewend was.

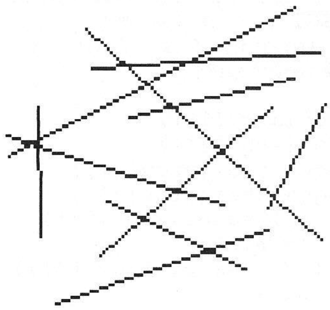
Line

Het meest simpele programma van de drie! *Line* zorgt er voor dat er een lijn wordt getekend. U geeft de start-x en -y posities en de bestemming-x en -y posities op en de lijn wordt getekend. Hoe luidt de syntax van *Line:*

LINE start-x start-y bestemming-x bestemming-y [kleur]

Geen opties of iets van dien aard.

Het is niet noodzakelijk de kleur op te geven, het commando zal standaard de kleur in register 1 gebruiken.



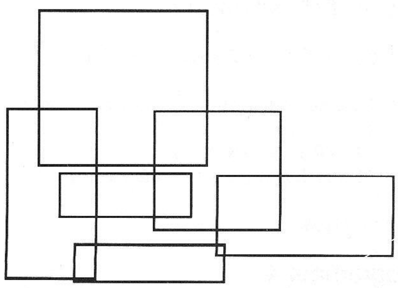
Treedt er een fout in het programma op dan zal de WARN (=5) flag gezet worden. Een voorbeeld:
LINE 10 10 300 100 2

Er zal een lijn getrokken worden van 10,10 naar 300,100 die de kleur zal hebben zoals is opgegeven in register 2.

Rect

Ook hier spreekt de naam voor zichzelf. Het commando zal naar keuze een niet of wel ingevuld rechthoek op het scherm toveren. Geef de x- en y positie op en de x- en y lengte en het commando geeft al resultaat. Overschrijding van de WorkBench schermmaten zal in een fout resulteren en ook hier zal bij een fout de WARN flag gezet worden.
Syntax van *Rect:*
RECT x y xlengte ylengte [fill] [kleur]

Het vullen van de rechthoek en de kleur zijn hier optioneel. Als u een gevulde rechthoek wilt geeft u 1 op, een niet gevulde rechthoek verschijnt als een 0 hier invult. U hoeft niets in te vullen. Standaard gebruikt het commando NIET INVULLEN.
Als u de kleur op wilt geven dan zult u ook op moeten geven of de rechthoek ingevuld of niet ingevuld moet worden. De kleur zal door het commando standaard op de kleur in register 1 gezet worden.



Een voorbeeld van *Rect:*
RECT 50 10 450 100 1 2

Het voorbeeld zal resulteren in een rechthoek beginnende op positie 50,10 en zal een lengte hebben van 450. De breedte zal in dit voorbeeld 100 pixels bedragen. Ook de invulparameter en de kleurparameter zijn hier ingevuld. De rechthoek zal ingevuld worden en het geheel zal de kleur hebben die opgegeven is in kleurregister 2.

Circle

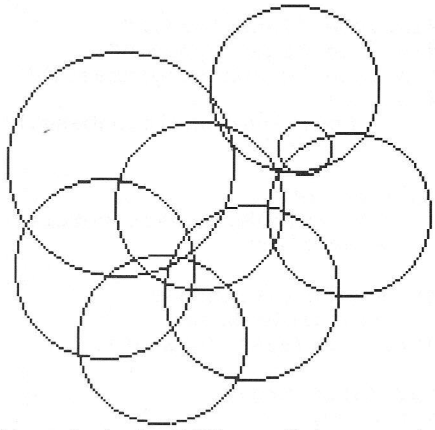
Tot slot het laatste nieuwe DOS commando. Het commando Circle zal een circel op het beeldscherm tekenen. U geeft de x- en y positie van het centrum van de circel op en u geeft de straal op. Het commando zal de gewenste circel dan voor u tekenen.
De syntax van *Circle:*
CIRCLE x y straal [stap] [kleur] [aspect]

De x-, y positie en straal zullen voor u duidelijk zijn. De waarde die u voor stap invult, bepaald hoe nauwkeurig en daarmee ook hoe snel, de circel getekent wordt. Standaard zal de waarde 10 gebruikt worden. Elke waarde groter zal voor een hogere tekensnelheid zorgen. De circel zal tevens meer en meer afwijken van de circelvorm. De mogelijkheid bestaat zelfs om een driehoek te tekenen! Elke waarde kleiner zorgt voor een nauwkeurigere circel doch u zult het verschil nauwelijks of niet kunnen zien. De snelheid zal met de kleinere waarden wel drastisch afnemen. Dus kleinere waarden zijn eigenlijk niet nodig.
De kleur zal ondertussen, gezien de vorige twee commando's, duidelijk zijn. Als laatste is er dan nog de aspect ratio. Zoals u weet is het DOS-scherm 640 pixels breed en maximaal 256 pixels hoog. Doordat de pixels hoger zijn dan ze breed zijn zal de circel niet rond worden. Vandaar dat er een aspect ratio wordt gebruikt om de circel te veranderen. Het commando gebruikt standaard de waarde 2.1. Een waarde kleiner zal zorgen dat er een smallere circel ontstaat, een grotere waarde dan 2.1 zal voor een meer afgeplatte circel zorgen.
Een voorbeeld:
CIRCLE 300 100 50 10 2 2.5

Hierdoor zal een circel met een middelpunt op de coordinaten 300,100 en een straal van 50 getekend worden. De stapgrootte is 10. De circel zal getekend worden in de kleur die opgeslagen is in register 2 en zal door de aspect van 2.5 lichtelijk afgeplat zijn.
Een waarschuwing is op z'n plaats. Het commando controleert niet of de circel buiten de grenzen van het scherm getekent wordt. Is dit het geval, dan is het niet denkbeeldig dat de computer gaat 'hangen'.

Epiloog

Het einde! We hebben alles nu verteld binnen het kader van deze cursus. Dit betekent echter niet dat Commodore INFO niets meer aan AmigaDOS gaat doen, integendeel. Nog vele artikelen zullen volgen. Met uw medewerking zullen we vele nieuwe DOS commando's publiceren in de Tips en trucs hoek.
Wat het echter wel betekent is dat de serie



Binnenin AmigaDOS, met dit laatste artikel als slot, definitief tot het verleden gaat behoren.
De laatste woorden zijn gesproken. Au revoir mes amis! May the force be with you, we'll meet again.....

Johan & johan.

Programma 1. Een simpel voorbeeld DOS commando.

Instructies.

Programmanaam	: simpeld.c
Programmeertaal	: C
Gebruikt programma	: Aztek C v3.6
Opties	: cc simpeld.c +L ln simpeld.o -lc32
Aanroep	: simpeld

```
#include "functions.h"
#include "exec/types.h"

main(argc,argv)
int argc;
char *argv[];
{
 register int i;

 printf("Aantal variabelen
    => %2d\n",argc);

 for(i=0;i<argc;i++)
   printf("Variabele %2d...
    luidt => %s\n",i,argv[i]);
}
```

Programma 2.
Grafisch DOS commando LINE

Instructies.

Programmanaam	: line.c
Programmeertaal	: C
Gebruikt programma	: Aztek C v3.6
Opties	: cc line.c +L ln line.o -lc32

Aanroep : line s_x s_y d_x d_y [kleur]

```
#include "functions.h"
#include "exec/types.h"
#include "graphics/gfxbase.h"
#include
    "intuition/intuitionbase.h"


#define RP
    IntuitionBase->ActiveScreen
    ->RastPort

struct IntuitionBase
    *IntuitionBase;
struct GfxBase *GfxBase;

main(argc,argv)
int argc;
char *argv[];
{
 SHORT x1,y1,x2,y2,kleur;

 if(!(IntuitionBase=(struct
    IntuitionBase *)

    OpenLibrary("intuition.libr
    ary",0L)))
     Quit(5,"Geen
    intuition.library!\n");
 if(!(GfxBase=(struct GfxBase
    *)

    OpenLibrary("graphics.libra
    ry",0L)))
     Quit(5,"Geen
    graphics.library!\n");

 if(argc==1)
  Quit(5,"Syntax van LINE:
   LINE x1 y1 x2 y2
   [kleur]\n");

 x1=atoi(argv[1]);
 y1=atoi(argv[2]);
 x2=atoi(argv[3]);
 y2=atoi(argv[4]);
 if(!(kleur=atoi(argv[5])))
    kleur=1;

 if(x1>0 && x1<640) && (x2>0
    && x2<640) &&
    (y1>0 && y1<256) && (y2>0
    && y2<256))
     {
      SetDrMd(&RP,JAM1);
      SetAPen(&RP,kleur);
      Move(&RP,x1,y1);
      Draw(&RP,x2,y2);
     }
 else
  Quit(5,"Workbench
   resoluties : 0<x<640,
   0<y<256\n");

 Quit(0,"");
}

Quit(how,why)
int how;
char *why;
{
 printf("%s",why);

 if(IntuitionBase!=NULL)
  CloseLibrary(IntuitionBase);
 if(GfxBase!=NULL)
  CloseLibary(GfxBase);

 exit(how);
}
```

Programma 3.
Grafisch DOS commando: RECT

Instructies.

Programmanaam	: rect.c
Programmeertaal	: C
Gebruikt programma	: Aztek C v3.6
Opties	: cc rect.c +L ln rect.o -lc32

Aanroep : rect x y xl yl [fill][kleur]

```
#include "functions.h"
#include "exec/types.h"
#include "graphics/gfxbase.h"
#include
  "graphics/gfxmacros.h"
#include
   "intuition/intuitionbase.h"

#define RP
   IntuitionBase->ActiveScreen
   ->RastPort

struct IntuitionBase
   *IntuitionBase;
struct GfxBase *GfxBase;
main(argc,argv)
int argc;
char *argv[];
{
 SHORT
   x,y,XLength,YLength,kleur;
 BOOL fill;

 if(!(IntuitionBase=(struct
   IntuitionBase *)
   OpenLibrary("intuition.libr
   ary",0L)))
     Quit(5,"Geen
   intuition.library!\n");
 if(!(GfxBase=(struct GfxBase
   *)

   OpenLibrary("graphics.libra
   ry",0L)))
     Quit(5,"Geen
   graphics.library!\n");

if(argc==1)
  Quit(5,"Syntax van RECT:
   RECT x y x_length
   y_length [fill=0 of 1]
   [kleur]\n");

x=atoi(argv[1]);
y=atoi(argv[2]);
XLength=atoi(argv[3]);
YLength=atoi(argv[4]);
if(!(fill=atoi(argv[5])))
   fill=FALSE;
if(!(kleur=atoi(argv[6])))
   kleur=1;

if((x>0 && x<640) &&
   (x+XLength>0 &&
   x+XLength<640) &&
    (y>0 && y<256) &&
   (y+YLength>0 &&
   y+YLength<640))
     {
      SetDrMd(&RP,JAM1);
      SetAPen(&RP,kleur);
      switch(fill)
       {
        case FALSE:
          Move(&RP,x,y);

    Draw(&RP,x+XLength,y);

    Draw(&RP,x+XLength,y+YLengt
    h);

    Draw(&RP,x,y+YLength);
          Draw(&RP,x,y);
          break;

        case TRUE:
          BNDRYOFF(&RP);

    RectFill(&RP,x,y,x+XLength,
    y+YLength);
          break;
       }
     }
  else
    Quit(5,"WorkBench
    resoluties: 0<x<640,
    0<y<256\n");

 Quit(0,"");
}

Quit(how,why)
int how;
char *why;
{
 printf("%s",why);

 if(IntuitionBase!=NULL)

    CloseLibrary(IntuitionBase)
    ;
 if(GfxBase!=NULL)
   CloseLibrary(GfxBase);

 exit(how);
}
```

Programma 4.
Grafisch DOS commando CIRCLE

Instructies.

```
Programmanaam        : circle.c
Programmeertaal      : C
Gebruikt programma   : Aztek C v3.6
Opties               : cc circle.c +L
                       ln circle.o -lm32 -lc32
Aanroep   : circle x y straal [stap]
                       [kleur][aspect]
```

```
#include "functions.h"
#include "math.h"
#include "exec/types.h"
#include "graphics/gfxbase.h"
#include
    "intuition/intuitionbase.h"

#define RP
    IntuitionBase->ActiveScreen
    ->RastPort
#define PI 22.0/7.0

struct IntuitionBase
    *IntuitionBase;
struct GfxBase *GfxBase;
DOUBLE atof();

main(argc,argv)
int argc;
char *argv[];
{
 SHORT x,y,straal,kleur;
 DOUBLE aspect;
 register int i,xn,yn;
if(!(IntuitionBase=(struct
   IntuitionBase *)

   OpenLibrary("intuition.libr
   ary",0L)))
     Quit(5,"Geen
   intuition.library!\n");
if(!(GfxBase=(struct GfxBase
   *)

   OpenLibrary("graphics.libra
   ry",0L)))
     Quit(5,"Geen
   graphics.library!\n");

if(argc==1)
   Quit(5,"Syntax van CIRCLE:
   CIRCLE x y straal [stap]
   [kleur] [aspect]\n");

x=atoi(argv[1]);
y=atoi(argv[2]);
straal=atoi(argv[3]);
if(!(stap=atoi(argv[4])))
   stap=10;
if(!(kleur=atoi(argv[5])))
   kleur=1;
if(!(aspect=atoi(argv[6])))
   aspect=2.1;

SetDrMd(&RP,JAM1);
SetAPen(&RP,kleur);


   xn=x+aspect*(straal*cos(0.0
   ));
yn=y+(straal*sin(0.0));
 Move(&RP,xn,yn);

 for(i=0;i<361;i+=stap)
   {

     xn=x+aspect*(straal*cos(PI/
     180.0*(float)i));

     yn=y+(straal*sin(PI/180.0*(
     float)i));
      Draw(&RP,xn,yn);
    }
 Quit(0,"");
}

Quit(how,why)
int how;
char *why;
{
 printf("%s",why);

 if(IntuitionBase!=NULL)

     CloseLibrary(IntuitionBase)
     ;
 if(GfxBase!=NULL)
    CloseLibrary(GfxBase);

 exit(how);
}
```

Killers & bandieten

Tips en trucs voor de Amiga

Net een nieuwe CD gekocht. Klinkt wel lekker! Hmm, zo gaat je humeur ook weer stukken vooruit. Wist je dat psychologisch onderzoek aangewezen heeft dat de werkprestatie van mensen enorm vooruit gaat als je de juiste muziek draait! HAAAARRRR-RYYYY.... Goed, even to the point..

Tjonge jonge, een moordlustige kop dit keer. Hopelijk is de inhoud iets minder in die richting. Nee, geen handleiding tot virussen dit keer. By the way, nog last gehad van zwart uitslaande schermen, niet geheel naar uw zin functionerende computers en meer van dat soort gedonder? Druk eens 'ALT, COMMODORE, SPATIE, AMIGA en ALT. Functioneert de computer weer. Juist ja, een BYTE BANDIT. Even opnieuw installeren die schijf!!

PokeMania II

Nee, geen nieuwe poke's. Alleen even een kleine toevoeging op wat er in de vorige aflevering gedrukt is. Door enige kleine fout(en)jes, zijn we vergeten de naam van de inzender af te drukken. Dat volgt dus nu even. De inzender was, Martijn Schot uit Terschelling. Martijn, 'keep up the good work' en houd vooral niet op in te sturen!!

Screens op 80 graden Celcius

Geen paniek, uw monitor zal er niet van te lijden hebben. Geen brand in uw videoprocessor of wat voor catastrofale armageddon-achtige verschijnselen dan nog meer. Integendeel zelfs. Een bijzonder leuk grafisch grapje kregen we ingestuurd. In assembly dit keer. Gebruik de Seka assembler. Typ de source in en schrijf deze daarna weg met het W (lees Write) commando. Typ hierna A in, wat voor Assemble staat. Schrijf hierna de uitvoerbare file weg naar disk door middel van het WO (WriteObject) commando. Nu kunt u, zowel vanuit de Seka assembler als vanuit DOS, het commando uit voeren. Druk de 'G' toets, wat voor Go staat, waardoor het commando uitgevoerd wordt, of druk het uitroepteken '!' in waardoor u in DOS komt. Vanuit DOS typt u de naam in waaronder u het met WO heeft weggeschreven.
Genoeg uitgelegd, hier is de listing.

```
* CRUNCH SCREEN, SOURCE BY M.
  KOEL

ExecBase      = 4
OpenLibrary   = -408
ScrollRaster = -396

movem.l D0-D7/A0-A6,-(A7)
; registers redden
; Haal Graphics Base
move.l        ExecBase,A6
lea           GfxLib,A1
sr            OpenLibrary(A6)
move.l        D0,GfxBase
; Haal Intuition Base

    lea       IntLib,A1
    jsr       OpenLibrary(A6)
    move.l    D0,IntBase

    move.l      #126,D6
    move.l      GfxBase,A6

; herhaal 126 keer

loop:
    move.l    IntBase,A1
    move.l    52(A1),A1 ;haal
    rastport
    move.l      50(A1),A1
    move        #0,D0         ;scroll
    gebied
    move          #-1,D1
    move          #0,D2
    ;(0,0)-(639,126)
    move          #0,D3
    move          #639,D4
;0 pixels in x- en -1 in
    y-richting
    move          #126,D5
    jsr           ScrollRaster(A6)
    ;scroll

    move.l      IntBase,A1
    move.l      52(A1),A1 ;haal
    rastport
    move.l       50(A1),A1
     move          #0,D0
    ;scroll gebied
    move          #1,D1
     move          #0,D2
    ;(0,126)-(639,256)
    move          #126,D3
    move          #639,D4
    move          #256,D5
    jsr           ScrollRaster(A6)
;scroll naar boven, 0 in x
    en 1 in y

    dbra        D6,loop    ;en
    herhaal

    movem.l     (A7)+,D0-D7/A0-A6
;herstel registers
    rts

GfxLib: dc.b
    "graphics.library",0
IntLib: dc.b
    "intuition.library",0

even

GfxBase: dc.l 0
IntBase: dc.l 0
```

Door: Martin Koel uit Heerhugowaard.

Waarschuwing

In het vorige nummer van Commodore Info stond een POKE voor de Amiga:
`POKE 8977,34` = mouse-pointer verdwijnt;
`POKE 8977,12` = mouse-pointer verschijnt weer.

Als je niet wilt riskeren dat je de beruchte melding "software error" krijgt, laat deze truc dan achterwege!
Het genoemde effect doet zich namelijk inderdaad voor, maar een foutmelding, een Requester (bv "No disk present in unit ...") en de opdracht ON ERROR GOTO zijn dan levensgevaarlijk geworden.

Ferry Groothedde (Het Computerblad AMIGA)

Arkanoid Revenge of Doh

Voor de spelliefhebbers is deze. Het is maar een zeer korte doch wel leuke. Houd onder het laden de linkermuisknop ingedrukt. Wacht enige tijd. En zie wat er verschijnt!!!

Sword of Sodan

Kent u dit spel al. Een zeer fraai spel. Gigantisch goeie graphics en zeer fraaie samples en melodieën. Toch valt het bijzonder tegen om het spel te spelen. Voor je het weet wordt je goed onderuit gehaald en kun je de rest van het spel wel schudden want dat krijg je niet te zien. Ook hier is een oplossing voor. Oneindige levens heet het. Hoe deze te krijgen vraagt u zich nu af. Volg de volgende instructies.
Laad het spel en kies 'HERO'. Zorg er nu voor dat je al je levens verliest in het EERSTE scherm. Kies nu opnieuw, echter kies dan voor 'HEROINE'. Je dappere heldin zal nu onverslaanbaar zijn want ze beschikt over oneindige levens. Als je achter alle geheimen van het spel, vooral in het kasteel, wilt komen, dan zul je nu ongeveer 40 minuten nodig hebben om het spel te volbrengen. Succes!

Schone schermen

Werkt u veel in CLI en heeft u nogal eens last van een smerig (niet beeldbuissmerig!) scherm, let dan even op want de volgende tip is voor u.
Er zijn drie manieren om een CLI scherm schoon te krijgen.
1. ECHO "*ec". Het echo commando wordt geladen en scherm wordt schoon geveegd.
2. ALIAS clear ECHO "*E[0;0H*E[J". Dit is alleen toe te passen in de nieuwe SHELL van WorkBench 1.3 en wordt gewoonlijk al automatisch gedaan in de shell-startup routine die door het newshell commando automatisch uitgevoerd wordt. Typ 'clear' in en het ECHO commando wordt geladen.
3. Onze nieuwe manier. We hebben een kleine C routine geschreven (piepklein en wordt toch oh zo groot!) welke het ASCII teken 12 afprint. Deze zorgt ervoor dat het besturingssysteem het beeldscherm schoon maakt.
Hier volgt het listinkje.

```
main()
{
 printf("%c",12);
}
```

Dat is alles, klein maar fijn zullen we maar zeggen. Compile en link het onder de naam 'cls.c' en 'cls.o' en zet het uitvoerbare bestand 'cls' in uw c-directory. Als u voortaan 'cls' intypt zal het scherm gewist worden. O ik wordt er zo ziek van

Listing bij Move in DOS

```
#include "stdio.h"
#include "functions.h"
#include "exec/types.h"

#define EEN 1
#define END 255

Quit(how,why)
int how;
char *why;
{
 printf("%s",why);
 exit(how);
}
main(argc,argv)
int argc;
char *argv[];
{
 FILE *File1,*File2,*fopen();
 register int i,j;
 UBYTE FileRNaam[30],FileWNaam[30];

 if(argc <= EEN || strcmp("?",argv[1])==NULL)
  Quit(5,"Syntax MOVE: Move drive:file [TO] drive:[file]\n");

 strcpy(FileRNaam,argv[1]);

 i=j=NULL;
 while(i<30 && FileRNaam[i++]!=':');
 if(i<30)
   for(j=i;j<30;j++)
     FileRNaam[j-i]=FileRNaam[j];

 if(!(File1=fopen(argv[1],"r")))
   Quit(5,"File is niet aanwezig!\n");

 if(strcmp("TO",argv[2])!=NULL && strcmp("to",argv[2])!=NULL)
  strcpy(FileWNaam,argv[2]);
 else
  strcpy(FileWNaam,argv[3]);

 i=j=NULL;
 while(i<30 && FileWNaam[i++]!=':');
 if(i>30)
   Quit(5,"De bestemming dient van een DRIVENAAM te worden
    voorzien\n");

 if(FileWNaam[i]<32 || FileWNaam[i]>123)
   for(j=0;j<(30-i);j++)
     FileWNaam[i+j]=FileRNaam[j];

 File2=fopen(FileWNaam,"w");

 ch=getc(File1);
 do {
  putc(ch,File2);
  ch=getc(File1);
 }while(ch!=END && ch!=EOF);

 fclose(File1);
 fclose(File2);
 if((DeleteFile(argv[1]))==NULL)
 {
    DeleteFile(FileWNaam);
    Quit(5,"Het originele bestand is niet te
   verwijderen!\n");
 }
}
```

Listing bij Fractals

```
#include "math.h"
#include "stdio.h"
#include "functions.h"
#include "exec/types.h"
#include "graphics/gfx.h"
#include "graphics/gfxmacros.h"
#include "intuition/intuition.h"

struct Screen *Screen;
struct IntuitionBase *IntuitionBase;
struct GfxBase *GfxBase;
struct NewScreen NewScreen = {
    0,0,320,256,5,
    0x00,0x01,NULL,
    CUSTOMSCREEN,NULL,
    (UBYTE *)"Fractals",
    NULL,NULL };

UWORD Colors[] =
{
 0x0008,0x0060,0x0ff0,0x0fd0,0x0fb0,
 0x0f90,0x0f90,0x0f70,
 0x0f70,0x0f70,0x0f40,0x0f20,0x0f00,
 0x0f01,0x0f03,0x0f05,
 0x0f06,0x0f07,0x0f08,0x0f09,0x0f0a,
 0x0f0b,0x0f0d,0x0f0f,
 0x0d0f,0x0b0f,0x090f,0x070f,0x050f,
 0x000f,0x000d,0x000a,
};
int xres,yres;
char *MouseL = (char *)0xbfe001;

main()
{
 DOUBLE
    AngleR,AngleI,Side,DistanceX,DistanceY;
 DOUBLE CR,CI,A,B,ZI,ZR;
 DOUBLE Length;
 register int x,y,Iteration;

 AngleR=-.70;
 AngleI=-1.25;
 Side=.80;
 xres=yres=1;
 printf("\nWaarde voor AngleR, AngleI, Side,
    xres, yres : ");
 scanf("%f%f%f%d%d",&AngleR,&AngleI,
    &Side,&xres,&yres);

 OpenAll();
 LoadRGB4(&Screen->ViewPort,Colors,32);
 RemakeDisplay();
 DrColBrdr();

 CR=CI=ZR=ZI=A=B=Length=0.00;

 DistanceX=Side/200.0;
 DistanceY=Side/200.0;

 for(y=0;y<=200;y+=yres)
   {
    for(x=0;x<=200;x+=xres)
      {
       CR=(DOUBLE)x*DistanceX+AngleR;
       CI=(DOUBLE)y*DistanceY+AngleI;
       ZR=CR;
       ZI=CI;
       Iteration=0;
       Length=0.00;
       while(Length<3.00 && Iteration < 33)
         {
          A=ZR*ZR;
          B=ZI*ZI;
          Length=sqrt(A+B);
          ZI=2*ZR*ZI+CI;
          ZR=A-B+CR;
          Iteration++;
         }
       Teken(x,y,Iteration);
      }
   }
 while((*MouseL & 0x40)==0x40);

 CloseScreen(Screen);
 CloseLibrary(GfxBase);
 CloseLibrary(IntuitionBase);
}
Teken(xx,yy,kleur)
int xx,yy,kleur;
{
 SetAPen(&Screen->RastPort,kleur);
 BNDRYOFF(&Screen->RastPort);
 RectFill(&Screen->RastPort,
   (xx+30-(xres/2)),(yy+15-(yres/2)),
   (xx+30+(xres/2)),(yy+15+(yres/2)));
}
OpenAll()
{
 if(!(IntuitionBase=(struct IntuitionBase
    *)OpenLibrary("intuition.library",0L)))
   {
    printf("Geen intuition.library\n");
    exit(5);
   }
 if(!(GfxBase=(struct GfxBase
    *)OpenLibrary("graphics.library",0L)))
   {
    printf("Geen graphics.library\n");
    CloseLibrary(IntuitionBase);
    exit(5);
   }
 if(!(Screen=(struct Screen *)
   OpenScreen(&NewScreen)))
     {
      printf("Scherm wil niet!\n");
      CloseLibrary(GfxBase);
      CloseLibrary(IntuitionBase);
      exit(5);
     }
}

DrColBrdr()
{
 register int i;

 BNDROFF(&Screen->RastPort);
 for(i=0;i<33;i++)
     {
      SetAPen(&Screen->RastPort,i);
      RectFill(&Screen->RastPort,250,
             30+(i*5),260,35+(i*5));
     }
}
```

he, waarom nu pas. Oke beter laat dan NOOIT nietwaar!

Move in DOS

Weer eens een 'nieuw' DOS commando. Overkomt het u niet vaak dat u een file van de RAM disk copieert naar drive df0: waarna u dan de file van de RAM disk moet wissen? Hiervoor hebben we een oplossing gevonden. We introduceren hier een 'nieuw' DOS commando die de file copieert naar een opgegeven bestemming en vervolgens de originele file wist. Het commando heet Move. Hoe luidt de syntax van Move?

```
MOVE drive:filenaam [TO]
    drive:[filenaam]
```

Zoals u ziet mag u het keyword TO gebruiken, het is echter niet noodzakelijk. Ook de bestemmingsfilenaam kunt u achterwege laten. Deze wordt dan overgenomen van de eerste vermelding. De bestemmingsdrive is wel noodzakelijk. Move ondersteund nog geen wildcards dus verwonder u zelf er niet over als de melding 'File is niet aanwezig' verschijnt. Typ de nevenstaande listing in. Compile het met de optie +L en link het met de library 'c32.lib'.

Fractals

We hebben bijna een traditie van grafische grapjes opgebouwd. In elke aflevering introduceren we er wel één. Deze keer zijn het er zelfs twee geworden. Zo kunt u bovenaan in deze aflevering een schermcruncher vinden, en hier een beetje uit de hand gelopene. Het programma tekent hele simpele fractal figuren. Niet driedimensionaal, wel mooie kleuren. Er valt eigenlijk niet veel van te zeggen. Een kleine opmerking nog wel. Bij het invullen van de getallen moet u erop letten dat de getallen gescheiden worden door spaties. Als voorbeeld geven we "-2 -1.25 2.5 1 1". Vul dit zo in, onthoud u mag geen komma's e.d gebruiken, gewoon tussen elk getal een spatie. Dit voorbeeld zal dan een fractal in z'n meest algemene vorm afbeelden.

Rectificaties

We blijven ons er voor schamen. Toch kunnen wij er ook niet veel aan doen. Fouten in programma's. De eerste zat in het listinkje behorende bij Rnd in C uit de C_Info nr 6, blz. 72. De subroutine hoort te luiden;

```
float rnd(x)
int x;
{
UBYTE *Seed=(UBYTE *)0xbfd800;
return(((float)*Seed/255.0));
}
```

Het volgende foutje troffen we aan in de listing van het programma 'loadilbm.c', C_Info nr.6, blz. 81. De opties waren niet helemaal correct. Ze horen te luiden:

```
cc loadilbm.c +L
ln loadilbm.o -lc32
```

Tot slot

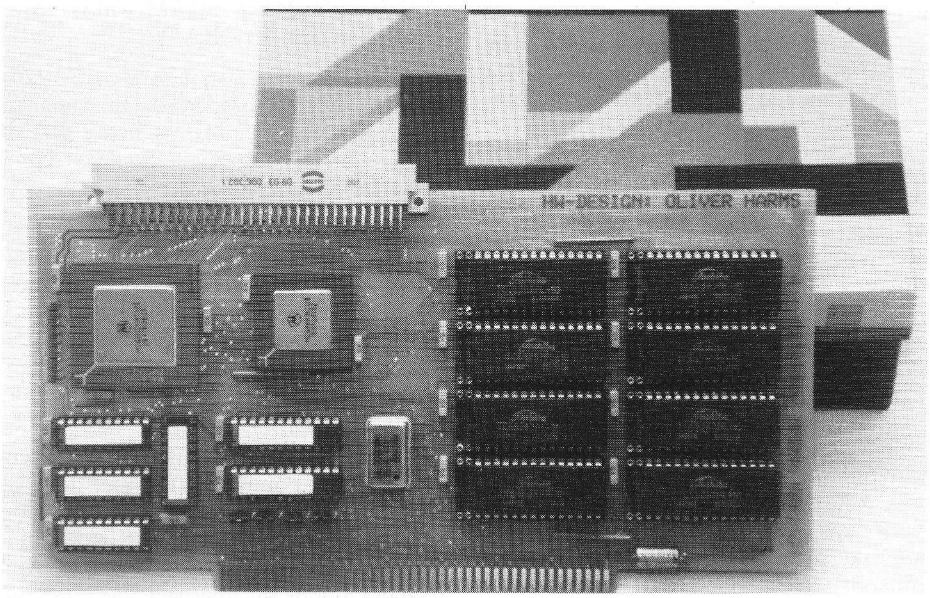
Zo, tot zover weer ons verhaal. Kost iedere keer meer moeite om de tips en trucs bij elkaar te krijgen. Kom op mensen, we hebben ze echt nodig! Zonder uw medewerking weten we niet wat u interesseert en kunnen we ook moeilijk wat in elkaar knutselen. Dus, massaal reageren. Heeft u vragen, opmerkingen, tips en trucs, alles is welkom!!!!
Dan voor nu welterusten en morgen vroeg weer op. Pggfoooeoaaw, nee dus! (niet vroeg graag, 't is weekend morgen, bij ons dan tenminste)

Johan & johan

Turboboards voor de Amiga

Veel snelheid voor veel geld

In PC-snelheidstermen gezien is de Motorola 68000 CPU wat bejaard aan het worden. Bij de Commodore Amiga-lijn draait deze 16/32-bits microprocessor op 7.14 MHz en dat is zelfs voor een AT een beetje aan de trage kant. Tegenwoordig heeft de doorsnee business-class IBM PC compatibele PC en kloksnelheid van 12 tot 16 MHz of meer. Bezitters van de Amiga 2000 (en bij enkele boards ook de Amiga 500) kunnen deze Commodore-telg eenvoudig opvoeren. Een turbokaart met Motorola MC 68020 en de floating-point coprocessor MC 68881 in een uitbreidingsslot steken en de Amiga scheurt op 14 MHz verder. Wie nog meer wil kan zelfs een MC 68030-kaart op 25 MHz laten razen.



Snelheid lijkt soms wel het grootste goed in computerland. Hoe sneller hoe beter en des te meer de bezitter kickt op zijn/haar machine. Nu is snelheid altijd een ingewikkeld samenstelsel van verschillende componenten. Van invloed zijn o.a. de kloksnelheid van de CPU, de ontlastende hulp die de coprocessoren bieden, de bus (16 of 32 bits), de snelheid van de drives en of de machine onder multi-tasking draait. Wat de snelheid van de Motorola microprocessoren betreft zijn er twee mogelijkheden voor meer snelheid: 1.). Koop een nieuwe Amiga 2500 of 3000 en 2). Zet een turboboard in de bestaande Amiga 2000 of 500. Deze laatste optie kost een lieve duit, maar blijft natuurlijk wel aanmerkelijk goedkoper dan een dure 2500- of 3000-machine te kopen. Wij bekeken een aantal van de Amiga MC 68020- en MC 68030-turboboards voor u.

De hamvraag bij het opvoeren van de snelheid blijft altijd: "Is het nu allemaal wel echt nodig?". Het antwoord op deze vraag kan per gebruiker flink verschillen. Om het eerlijk te zeggen, de hobbyist die niet met ingewikkelde grafische of animatiepakketten werkt, geen grootschalige desktoppublishing bedrijft of veel gecompliceerd rekenwerk verricht heeft meer dan voldoende aan de standaard MC 68000. De Amiga 2000 en 500 zijn nog altijd niet echt verouderd en kunnen bij vele hobbyisten en gewoon kantoorgebruik goed meekomen

Anders wordt het als er professionele RES graphics, hoge school-animaties, ray-tracing, desktoppublishing van formaat, CAD/CAM, krachtige multi-tasking en het doorrekenen van grote spreadsheets of het bewerken van omvangrijke databases op het verlanglijstje staan. Dan kan die oude 7.14 MHz Motorola 68000 CPU te kort schieten. Een *turbo-* of *accelerator-board* kan de machine aan de gestelde hogere snelheideisen aanpassen. Dat kan op drie manieren:

- De 7.14 MHz MC 68000 vervangen door een *14 MHz MC 68020* of een *25 MHz CPU*
- Het inschakelen van een *floating-point coprocessor*, de MC 68881 of MC 68882
- Het verbeteren van de *systeemhardware*. Hieronder vallen zaken zoals het gebruik van snellere (Static) RAM-chips, cache- RAMS, het gebruik van een 32-bits in plaats van een 16-bits bus en de toepassing van een supersnelle harddisk met controller

De meeste turboboards gaan uit van de combinatie snellere CPU met floating-point coprocessor. Een aantal accellerator-boards past daarnaast ook supersnelle RAM- en cache-technieken toe.

Werking van een turboboard

Bij de installatie van een acceleratorboard in een uitbreidings slot of via een uitbreidingsbus neemt de snellere MC 68020 of 68030 de taken van de oude MC 68000 over. In het ene geval dient men daartoe de oude CPU te verwijderen. Bij een andere turbokaart kunt u de MC 68000 gewoon laten zitten. De versnellerkaart zet de MC 68000 gewoon op non-actief.

Dat klinkt allemaal gemakkelijker dan het in de praktijk is. Het vervangen van een CPU bij een IBM compatibele PC leidt al gauw tot herziening van het complete hoofdkaartontwerp. Gelukkig werd bij de Amiga 2000- en Amiga 500-architectuur al rekening gehouden met toekomstige uitbreidingen en opvoering. Dat betekent echter niet dat een simpel "chipje-verwisselen" voldoende is om uw Amiga door de geluidsbarriere te laten breken. Extra ondersteunende hardware en in vele gevallen ook software (tegenwoordig zit er vaak al een extra diskette voor de MC 68020- en/of MC 68881- Amigaversie in de doos) zijn onmisbaar voor een goede werking van het systeem.

Neem nu eens de MC 68020 CPU. Deze op 14,3 MHz geklokte Motorola microprocessor heeft een twee maal zo hoge kloksnelheid als de 7,14 MHz MC 68000. Daar blijft het echter niet bij, want dit rekenwondertje heeft ook nog eens een 256 byte instruction-cache voor complete "in-chip" programmaloops, 32-bits vermenigvuldigings- en deel-operaties, een 32-bits RAM-bus en communicatiepoorten voor coprocessoren aan boord. Dat betekent dat deze MC 68020 CPU veel kan mits de microprocessor de *juiste ondersteuning van coprocessoren* krijgt. Geijkte uitbreidingen op een turbokaart zijn dan ook een MC 68881 floating-point- en *M*emory *M*anagement *U*nit-coprocessoren.

De *floating-point coprocessor* is in feite niets anders dan een supersnelle simpele rekenaar. De drijvende-komma-coprocessor kan in pijplijnvorm een groot aantal identieke rekenstapjes bliksemsnel en zeer efficiënt uitvoeren. Dergelijke eenvoudige rekentaken zijn eigenlijk te min voor de Centrale Processing Unit, maar kosten wel veel processortijd. Door de CPU middels een floating-point mathematische coprocessor te ontlasten houdt deze chip meer tijd over voor andere softwaretaken en neemt de systeemsnelheid navenant toe. Met name CAD-software, spreadsheets, 3D-graphics-pakketten, animatieprogramma's en andere mathematische software maken gebruik van deze coprocessoren. Bij tekstverwerking en spelletjes hebben zij geen enkele zin.

De *MMU* wordt gebruikt om routines in het snelle RAM te zetten. Bijvoorbeeld de Kickstart-routines uit het relatief trage ROM naar het fast Static RAM te verplaatsen.

De von Neumann-bottleneck is ook bij de turbokaarten een duur probleem. Een CPU blijkt veelal sneller te rekenen dan het RAM de data kan aanleveren en weer opnemen. Dat impliceert wachttijden van de CPU op de RAM-datastromen en verpilling van systeemkracht. Aan die flessehals van von Neumann valt het volgende te doen:

- Het gebruik van *Static RAM-banken* die sneller zijn dan het conventionele Dynamic RAM. De ververstijden (refresh rates) van SRAM liggen ver beneden de 100 nanoseconde, terwijl die van het DRAM er meestal flink boven liggen. De CPU kan dus per tijdseenheid het SRAM vaker aanspreken dan het DRAM
- De toepassing van een *real 32-bits RAM-bus*. Dit bredere datapad maakt het kontakt met het on board RAM twee maal zo snel als over een conventionele 16-bits bus
- Het gebruik van een *cache-RAM*. De MC 68020 en MC 68030 hebben al een groot cache-RAM aan boord zodat grote veel gebruikte programmaroutines onder handbereik van de CPU blijven. Tijdrovend laden uit het conventionele RAM of van de harddisk/diskdrive is dan niet meer nodig

Al die hardwareuitbreiding stellen hoge eisen aan de compatibiliteit met de systeemhardware en Amiga-software. Gelukkig blijkt het dankzij de goede door Commodore vrijgegeven Amiga systeemdocumentatie allemaal best mee te vallen. Bovendien blijft bij de meeste turboboards de oude MC 68000 gewoon zitten en kunt u in geval van nood weer op deze originele CPU terugvallen!

De snelheidswinst

Of u door installatie van een turboboard ook inderdaad verbluffende snelheidsprestaties krijgt hangt sterk van de draaiende software en de ondersteunende hardware af. In theorie moet een MC 68020 met MC 68881-combimatie met gemak een factor 4 sneller halen. Bij een MC 68030 en MC 68882 ligt zelfs een factor 12 in het verschiet.

Veel blijft echter afhankelijk van het feit of de programmaroutines op de geboden faciliteiten aansluiten. Alleen die pakketten die daadwerkelijk voor een mathematische coprocessor en MC 680X0 CPU's geschreven zijn trekken daar ook optimaal profijt van.

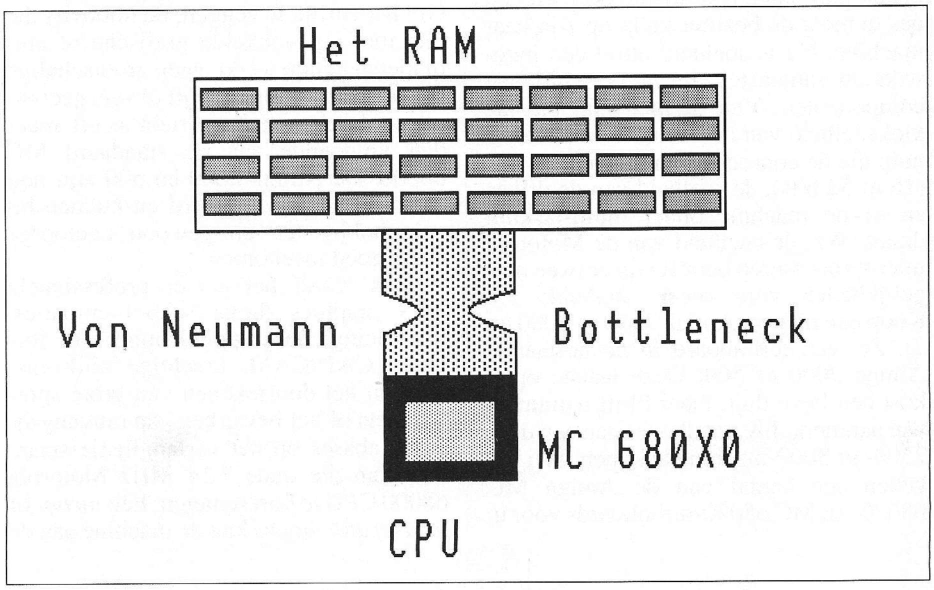
Een supersnel rekenteam met trage hardware-ondersteuning blijft altijd een reus op krukken. Alleen in combinatie met een snelle harddisk plus bijbehorende controllerkaart en 32-bits 70 ns (of sneller) SRAM-SIMMs valt alles uit de rekenkracht te halen wat er in zit. En dat maakt deze kaarten nu juist zo prijzig. Snelle harddisks en 32-bits SRAM is allerminst goedkoop. De prestaties zijn er echter wel naar en het zal zonde zijn u toch al dure turbokaart er niet mee uit te rusten.

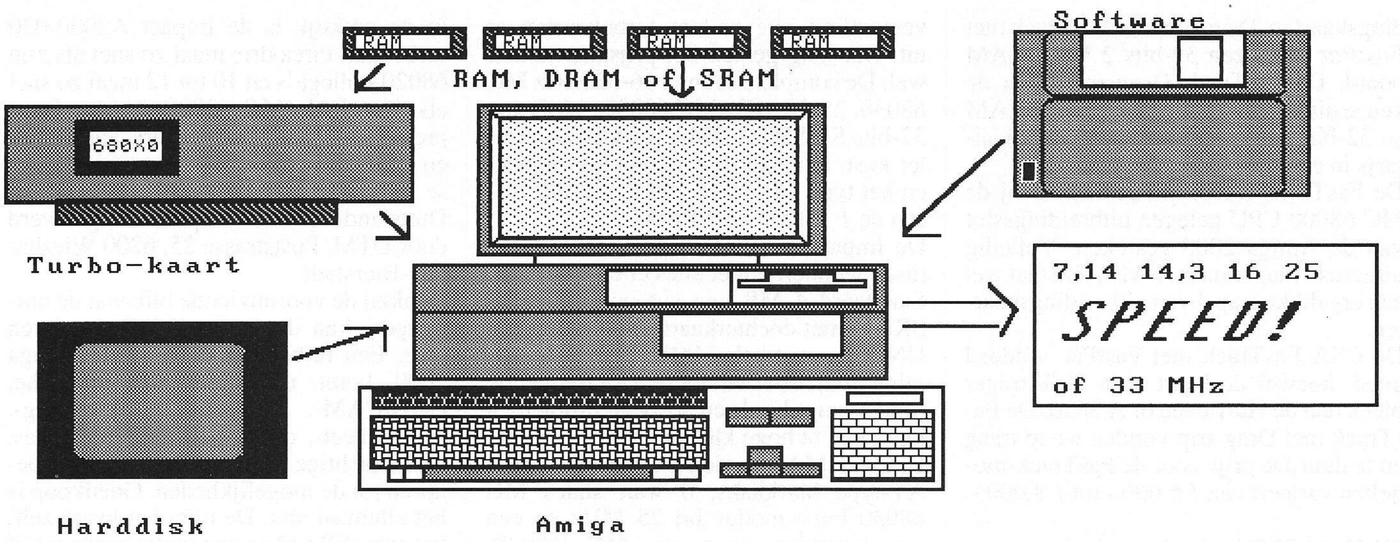
De MC 68020-turboboards

Commodore levert zelf de *A 2620*-uitbreidingskaart met 14,3 MHz MC 68020 en MC 68881 aan boord. De kloksnelheid van de mathematische coprocessor kan door keuze van het kwartskristal van 14,3 tot 25 MHz gevarieerd worden.

De taken van het geheugenmanagement berusten bij de MC 68851 MMU. Voor het draaien onder het UNIX multitaking-OS is deze MMU heel geschikt en zit dan ook standaard in de Amiga 2500 voor UNIX- gebruik.

De A 2620 wordt standaard met 2 MB aan boord geleverd. Op de kaart die wij onder





De systeemsnelheid hangt van vele factoren af

ogen kregen zaten 100 ns 32-bits RAM-chips. Beter zou zijn hier 80 ns SRAM-chips op te zetten. Dat kost natuurlijk weer extra. Uitbreiding tot 4 MB is mogelijk, maar vereist wel enige vaardigheid met technisch solderen. Installatie door de vakman verdient hierbij aanbeveling.
De inbouw van de kaart in de Amiga 2000 verschilt voor de modellen A (oude MC 68000 verwijderen) en B. Helaas was de meegeleverde duitse (de engelse is beter) gebruiksaanwijzing niet al te sterk en ook hier raden we u ook weer aan de turbokaart door een vakman/vrouw in de Amiga te laten zetten. Dat voorkomt veel gekluns en een mogelijke beschadiging van de electronica.
De snelheidsprestaties van de Commodore Amiga A 2620 turbokaart zijn zondermeer goed te noemen. Gemiddeld twee tot drie maal sneller als de MC 6800-uitvoering van de Amiga 2000. Makkelijk is de autoconfiguratie van het RAM dat geheel automatisch verloopt. Een tweede pluspunt is de accessmogelijkheid voor DMA-devices. Met name in combinatie met Direct Memory Access harddiskcontrollers kunnen de systeemprestaties van de harde schijf flink toenemen. Kortom een goede turbokaartkeuze. De prijs ligt afhankelijk van de leverancier en uitrusting voor de standaardversie iets boven de *f* 4.000,-.
De *Hurricane* is inmiddels een begrip in de Amiga-wereld geworden. Dit "wervelstorm" turboboard is de snelste van de hier besproken MC 68020-acceleratorkaarten. De kaart zelf bevat naast de 14,32 MHz MC 68020 en 16 MHz MC 68881 nog wat extra video-electronica en ruimte voor 2-16 MB aan on board RAM-geheugen. De oude MC 68000 CPU kan bij de nieuwere Hurricane-versies rustig op zijn socket blijven zitten. U kunt nu gewoon tussen beide microprocessoren heen-en-weer schakelen.
Zoals reeds werd opgemerkt is de Hurricane een waar snelheidsmonster. De kaart is op enkele punten iets sneller dan de Commodore A 2620 en de mathematische MC 68881 kan tot 25 MHz opgevoerd worden. Er zijn zero wait states. Ook is het mogelijk om een MC 68882 op de wervelstorm te monteren.
Lastig is dat al het extra fast RAM niet autoconfigurerend is. Dat betekent dat u de configuratieroutine op een of andere wijze aan de startup-sequentie zult moeten koppelen. Dat is bij de ene startup wat moeilijker dan bij de andere.
De Ronin Hurricane H2000 is alleen geschikt voor de Amiga 2000. Het turboboard plus geheugen-dochterkaart pakt nogal dik uit hetgeen problemen kan geven bij de installatie van een hardcard. Via een piggyback board kan de MC 68020 door een MC 68030 CPU vervangen worden. Verder een prima kaart voor rond de *f* 4.000,-.
De *Animate Turboboards* zijn er voor de Amiga 2000, Amiga 1000 en Amiga 500. Voor de Amiga 500 en 1000 heeft u de Animate Turboboards I of II nodig. Beide turbokaarten beschikken over een 14,3 MHz 68020 CPU en een 14/16 MHz MC 68881 drijvende-komma mathematische coprocessor. Desgewenst kan men een MC 68882 inbouwen.
Allen het type II beschikt over een omschakelaar voor de MC 68000-modus. Via een kabeltje met schakelaar kunt u het in een socket gestoken board van buiten af omschakelen. Bij type I is het bij weigerende programmatuur niet mogelijk om op de MC 68000- modus over te schakelen!
In de praktijk blijken de Animate type I- en type II-turboboards qua snelheid wat bij de andere genoemde accelerators achter te blijven. Ondanks het efficiënte gebruik van het on board MC 68020-cache breekt de afwezigheid van 32-bits SRAM de versnellerkaart op. In de praktijk loopt de Amiga circa twee maal sneller dan de conventionele MC 68000-uitvoering. Daar tegenover staat wel de lage prijs, *f* 1.100,- voor type I en *f* 1.320,- voor type II.
Het Animate turboboard type III is de professionele Amiga 2000- uitvoering. De kaart wordt in een vrij slot gestoken en beschikt wel over 32-bits fast RAM. Naar keuze kan de gebruiker de mathematische coprocessor tot 33 MHz (MC 68881 en MC 68882) opvoeren. Via de perifere 32-bits adapter is uitbreiding in de vorm van een extra geheugenkaart of multifunctionkaart. Op de kaart zelf kunt u het SRAM van 128 KB tot 512 KB (wel een beperking) uitbreiden.
De type III kaart kost rond de *f* 3500,- en bleek in de praktijk iets langzamer dan de Hurricane en A 2620. Het installeren bleek bij alle drie de typen relatief eenvoudig.
Tot slot nog de amerikaanse *CSA CPU*-kaarten. Er zijn twee typen: de *FasTrack 2000* met FastPac en met Dragstrip. Beide kaarten beschikken over een 14,3 MHz MC 68020 CPU en naar keuze een MC 68881/68882 floatinng-point coprocessor. Via een piggyback board kan de MC 68020 door een MC 68030 vervangen worden.
Het verschil tussen de beide kaartmodellen zit hem in de 32-bits RAM-uitbrei-

dingskaarten. De peperdure FasTrack met *FastPac* biedt een 32-bits 2 MB SRAM board. De FasTrack *Dragstrip* biedt de keuze uit de toepassing van 16-bit DRAM en 32-bits SRAM. U heeft hier de totaalprijs in eigen hand.
De FasTrack wordt in het dichtst bij de MC 68000 CPU gelegen uitbreidingsslot van de Amiga 2000 gestoken. Volledig uitgerust, maxzimaal 16 MB, ontstaat wel een erg dikke stapel van uitbreidingskaarten.
De CSA FasTrack met FastPac voldeed goed, hoewel de kaart zo'n 20% trager bleek dan de Hurricane of A 2620. De FasTrack met Dragstrip vonden we te traag en te duur.De prijs voor de FasTrack-modellen varieert van *f* 5.000,- tot *f* 8.000,-.

GVP 68030 turbokaart

Onder de naam *Impact A2000-030* levert Great Valley Products sinds kort een MC 68030 turbokaart voor de Amiga 2000. Inderdaad kan hier van een ware "impact" gesproken worden, want deze accelerator kaart ramt met een factor 10-12 versnelling alle andere turbokaarten er uit. Dat mag gezien het prijskaartje ook wel. De complete set met 16-33 MHz MC 68030, 25-33 MHz MC 68882, 4 MB aan 32-bits SRAM en DMA harddiskcontroller kost, afhankelijk van de kloksnelheid en het type (70, 80 of 100 ns) RAM, tussen de *f* 7.000,- en *f* 8.000,-.
De Impact A2000-030 is een echt technisch wondertje met alles er op en er aan. Standaard 4 MB aan autoconfigurerend SRAM met dochterkaart-uitbreiding, een UNIX compatibele MMU, asynchrone timing met een variabele klokfrequentie (ondervangt hard- en softwareproblemen t.g.v. een te hoge kloksnelheid), 4 MegaBytes/sDMA harddisk-controller voor AT-type harddisks, 0 wait states MC 68030 burst-modus bij 25 MHz en een omschakelaar voor de MC 68030-MC68000-modus aan boord leveren een complete high- power Amiga binnen een bestaande Amiga 2000 af. Het meegeleverde SET CPU-programma verzorgt de optimale afstemming van de Centrale Processing Unit.
In de praktijk is de Impact A2000-030 turbokaart circa drie maal zo snel als zijn 68020-collega's en 10 tot 12 maal zo snel als de origele MC 68000 Amiga. Deze prestaties zijn uiteraard sterk applicatie- en hardware-afhankelijk. Een nederlandse importeur is ons nog niet bekend. In Duitsland wordt de Impact o.a. geleverd door DTM, Poststrasse 25, 6200 Wiesbaden-Bierstadt.
Dankzij de vooruitziende blik van de ontwerpers kan de Amiga 2000 nog jaren mee. Een turbokaart er in en de Amiga 2000 komt tot ongekende grafische, CAD/CAM-, animatie-, ray-tracing-, spreadsheet-, dtp- en databaseprestaties. Ook krachtige multitasking en UNIX behoren tot de mogelijkheden. Goedkoop is het allemaal niet. De turbohardware zelf, het extra SRAM en een snelle harde schijf kosten flink geld en de investering dient qua softwaregebruik natuurlijk te lonen. In afwachting van een goedkopere Amiga 3000 of 2500 het beste alternatief.
U.S.

De MC 680X0-familie

Al heel wat Motorola 680X0-telgen hebben het digitale levenslicht aanschouwd. Het originele ontwerp werd de laatste jaren sterk verbeterd en biedt nu alles wat voor een moderne PC noodzakelijk is. Hieronder een beknopt overzicht van de verschillende specificaties:

De MC 68000

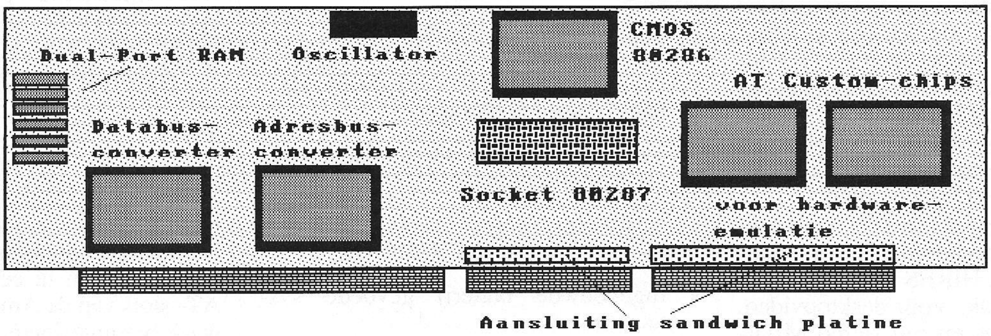
- **Klokfrequentie**; voor Amiga 7,14 MHz
- **Bus-breedte**: 16-bits systeembus, 32-bits RAM-bus
- **Direct Addressing Range**; 16 MB
- **Instruction types**: 56
- **Operations on 5 main datatypes**
- **Memory Mapped I/O**
- **Adressing modes**: 14
- **Data- en addresregisters**: 17

De MC 68020

- **Klokfrequentie**; voor Amiga 14,3/16,67 MHz
- **Bus-breedte**: 32-bits Non-Multiplexed systeembus, 32-bits RAM-bus
- **Dynamic bus-sizing**: 8-, 16- en 32-bits
- **Direct Addressing Range**: 4 Gigabyte
- **Operations on 7 main datatypes**
- **Memory Mapped I/O**
- **Addressing modes**: 18
- **Data- en addresregisters**: 16 voor 32-bits
- **On-chip cache**
- **Coprocessor-interfaces**; MC 68851 MMU, floating-point MC 68881 en MC 68882
- **Pijplijn-architectuur met interne parallel processing**
- **HCMOS-ontwerp**

De MC 68030

- **Klokfrequentie**; voor Amiga 16, 25 of 33 MHz
- **Bus-breedte**: 32-bits Non-Multiplexed systeembus, 32-bits RAM-bus
- **Dynamic bus-sizing**: 8-, 16- en 32-bits
- **Direct Addressing Range**: 4 Gigabyte
- **Operations on 7 main datatypes**
- On-chip **Paged MMU**
- **Addressing modes**: 18
- **Data- en addresregisters**: 16 voor 32-bits



De A 2286 AT-kaart

Een AT in de Amiga 2000

Het beste van twee werelden verenigd in één Amiga 2000. Aan de ene kant alle MC 680X0-kracht en aan de andere kant de complete MS-DOS-wereld van de IBM compatibele AT. Op het eerste gezicht een utopie. Bij nader inzien echter een realiteit met zo zijn beperkingen en kosten. In deze vergelijkende test van Commodore Info leest u of het werken met een tweekoppige Amiga loont

MS-DOS heeft het ondanks het oprukken van UNIX en OS/2 nog altijd voor het zeggen in de zakelijke PC-wereld. IBM PC compatibele AT's (Intel 80286) en 80386/486-klonen staan in vrijwel elk modern kantoor. De op de Motorola MC 680XO CPU gebaseerde PC's vormen een goede tweede bij de serieuze hobbyist en meer grafisch of video-georiënteerde zakelijke gebruiker. Voor beide systemen valt veel te zeggen. De IBM compatibele PC geeft toegang tot een schat aan redelijk geprijsde soft- en hardware. Bij de Amiga-lijn krijgt de gebruiker voor relatief weinig geld het summum aan PC grafische, animatie- en geluidsmogelijkheden in huis. Voor wie niet kunnen beslissen ligt de oplossing wellicht in de A 2286 AT-kaart die MS-DOS en Amigakracht in één machine verenigt.

De Amiga is al enkele jaren in staat MS-DOS programmatuur te draaien en ook IBM PC compatibele hardware te gebruiken. Als de wat teleurstellend verlopen pogingen met software-emulatie (Transformer) en de Sidecar voor de Amiga 1000 vergeten is het de Commodore A 2088 XT bridge-board (ook wel Januskaart genoemd) toch redelijk goed vergaan. Deze op de Intel 8088 (4,77 MHz en 8- bits systeembus) gebaseerde XT-brugkaart maakt het draaien van alle PC XT compatibele software en het aankoppelen van de overeenkomstige PC-hardware mogelijk. Dit alles tegen een allezins gunstige prijs. Zeker als men in aanmerking neemt dat uitbreiding met IBM PC harddisks en dergelijke goedkoper is dan bij de originele Amiga-hardware.

Jammer is dat Commodore met de A 2088 een beetje achter de feiten aanliep. Ten tijde van de introductie werkte weliswaar het gros van de PC-hobbyisten en de kleinere kantoorgebruikers nog met de PC XT. Nauwelijks een jaartje later was het allemaal al 10-16 MHz op de 16/32-bits bus AT wat de klok sloeg. Bovendien is een 4.77 MHz XT met 8-bits systeembus een slak in vergelijking met de 7,14 MHz Motorola MC 68000 met 16-bits bus. Een enkele leverancier ontdekte al een gat in de markt door een turbo-versie van de A 2088 XT-kaart te leveren. Bijvoorbeeld X-Pert Computer Service Idstein Duitsland vervangt de Intel 8088 voor de bekende NEC V20 8 MHz CPU en voor rond de 1.100 piek heeft u een XT die 80-100% sneller is dan het A 2088-origineel.

Toch maak je met een V20 nog geen AT van de Amiga. Daar is een compleet Amiga 2000-compatibele moederkaart voor nodig.

Waarom een AT?

De A 2286 is met een prijskaartje van rond de *f* 2.500,- (inclusief 1,2 MB AT diskdrive) niet goedkoop. Doe er nog *f* 500,- bij en je hebt een complete monochrome IBM compatibele PC met 40 ms 20 MB harde schijf. Waarom dan toch in de Amiga 2000 investeren?

In feite gaat het om twee vragen:

- 1). Is de AT-optie wel echt nodig?
- 2). Wat verdient de voorkeur, een Amiga AT of een losse IBM PC AT-kloon?

Eerst die *AT-optie.* Een IBM AT compatibele PC heeft een kloksnelheid van 8-25 MHz (afhankelijk van de CMOS CPU), een RAM- bereik van 16 MB, een 16-bits databus, en toegang tot zeer grote relatief gunstig geprijsde harddisks en veel PC-randapparatuur. Wat de besturingssyste-

men betreft is er keuze uit MS-DOS 3.X t/m 4.X, OS/2 en ook wel UNIX. De machine is een moderne kantoor-PC met software in alle maten en soorten. Databases, spreadsheets, dtp, tekstverwerking, CAD/CAM, het loopt allemaal zonder mankeren op de AT. Ook zijn er tegenwoordig vele goede WIMP-shells (Windows 286, GEM) die net als de Workbench voor de Amiga de gebruiksonvriendelijke bediening van de PC afschermen.
Een belangrijk verschil met de Amiga zit hem in de grafische, animatie- en muzikale mogelijkheden. HIRES kleur op de AT is een prijzige zaak. Voor desktopvideo, flitsende animaties en full stereo MIDI-aansturing zijn eveneens forse investeringen op de AT nodig. De Amiga 2000 is dan veel voordeliger. Kortom, het antwoord op vraag 1). luidt: Ja, als u een betrouwbaar en snel MS-DOS werkpaard nodig heeft.
Het antwoord op vraag 2). hangt sterk van de (toekomstige) toepassingen af. Wie het beste van beide besturingswerelden en/of inbouw van de goedkopere IBM PC-compatibele hardware in de Amiga 2000 wil kan het beste in de A 2286 investeren. Bovendien is het mogelijk om de grafische mogelijkheden van de Amiga voor MS-DOS software te benutten.

De A 2286-kaart

Qua specificaties voldoet de *A2286 bridge-board* precies aan de originele Advanced Technology, *AT*, -norm van IBM. Het oogt allemaal heel klassiek:

- Intel 80286 CPU
- Een kloksnelheid van 8 MHz
- Een 16-bits systeembus
- 1014 KB aan vrij RAM op de hoofdkaart
- Een 16 KB ROM-BIOS
- AT-busconnectoren voor de PC-slots van de Amiga 2000
- Vrij socket voor mathematische Intel 80287 coprocessor
- Aansluitingen voor 1.2 MB 5.25 inch en 3.5 inch 720 KB diskdrives
- Ingebouwde batterij gevoede systeemklok
- Set-upmenu met opslag gegevens in CMOS-chip
- DMA-optie

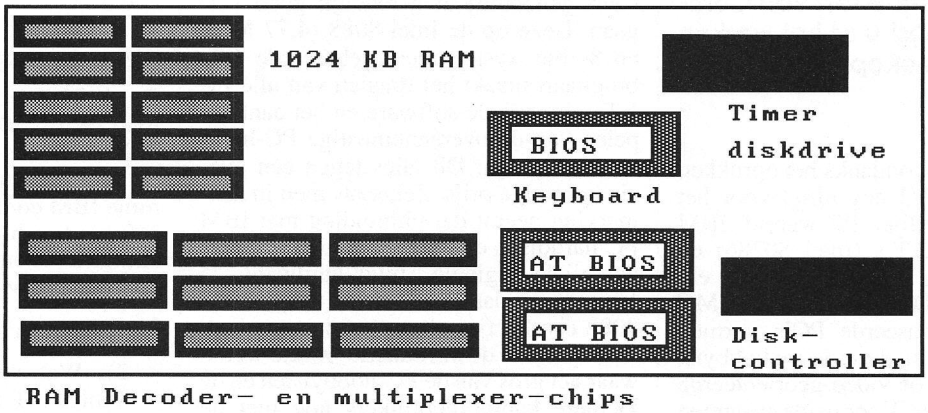
Schijn bedriegt echter. Het gaat hier wel degelijk om een geavanceerd stukje ontwerptechniek dat de MS-DOS PC-componenten exact afstemt op gebruik in een Amiga 2000-omgeving. Er zitten daartoe nogal wat custom-chips op de lange AT-kaart.
Het geheel bestaat uit de lange insteekkaart voor een AT-slot met dochterkaartje voor de AT-functies en 1024 KB aan vrij RAM. Op deze electronicakaarten vindt u:

- Een 64 KB Dual-Port-RAM voor communicatie tussen het Amiga- en PC-deel
- Emulatie van de IBM parallelle printerpoort op de Amiga-printerpoort
- Autoconfiguratie
- Video in monochroom of kleur, in MGA-, CGA- of Amiga- modus

Het werkt allemaal prima en als enige beperking geldt de wat trage 8 MHz snelheid van het MS-DOS-deel. De moderne AT draait 20-200 % sneller, maar dan spreken we ook wel over de iets duurdere klasse.

Installatie

Het *installeren* van de A 2286 brugkaart is tamelijk eenvoudig. U drukt de bridgeboard gewoon in één van de twee vrije AT- slots van de Amiga 2000 en de *hardware* is aangesloten. Er zijn dan nog één AT- en twee XT-slots over voor verdere PC- hardwareuitbreidingen. De meegeleverde 5.25 inch 1.2 MB diskdrive past in de drivebay onder die voor twee 3.5 inch diskettestations. Een standaard PC-lintkabel zorgt voor de datacommunicatie. De stroomvoorziening komt via één van de nog vrijliggende netstekkertjes. Het aansluiten van twee diskdrives is (voor alsnog) niet mogelijk.
De voor de installatie benodigde *software* staat op een standaard 3.5 inch Amiga-diskette. Met behulp van het op deze diskettte staande BRIDGE INSTALL dient de gebruiker een speciale A 2286 compatibele werkbankversie aan te maken. Met behulp van het aldus gemaakte AT.BOOT-bestand start de Amiga 2000 als hybride-PC op. Zowel het Amiga- als het IBM PC-deel kunnen dan gelijktijdig draaien. Verder zult u via het AT setupprogramma de datum, tijd, de hoeveelheid RAM en de drives moeten instellen. Is dat eenmaal gepiept dan heeft u er geen

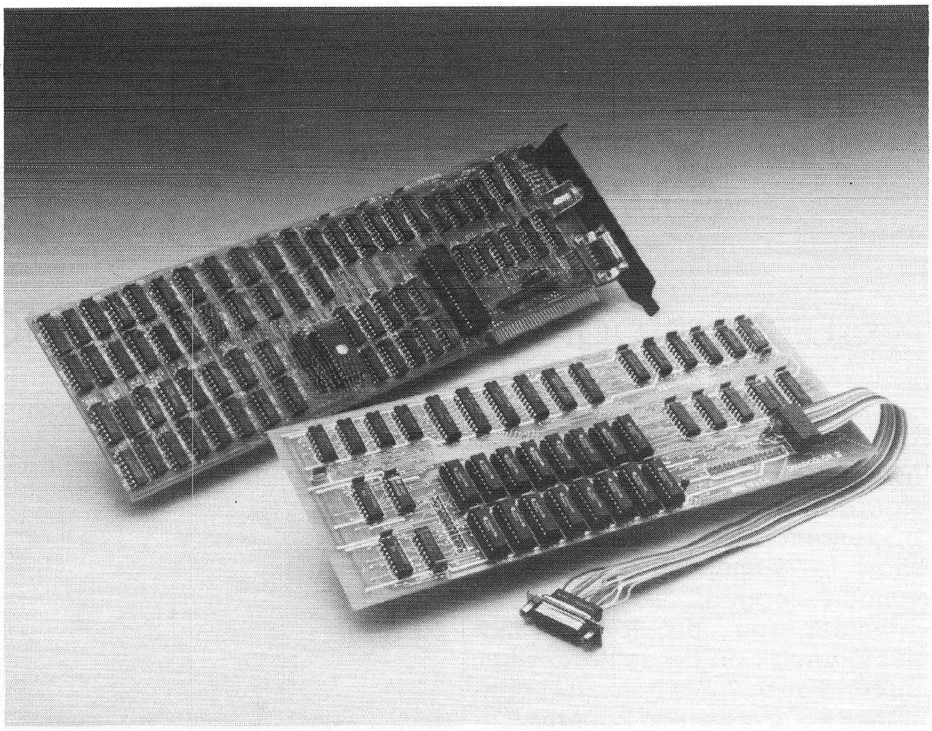


Schema dochterkaartje met de AT-controller en RAM

omkijken meer naar. Het AT- en keyboard BIOS-ROM, het Dual-Port RAM en de beide besturingssystemen doen de rest.

Het Dual-Port RAM

Ten einde de communicatie tussen de Amiga-DOS- en MS-DOS-wereld te bewerkstelligen hebben de Amiga-ontwerpers het *Dual-Port RAM* ontwikkeld. Dit 128 Kb metende geheugenbereik is een extra RAM- bank op de brugkaart. Beide computerdelen kunnen naar dit aparte geheugen bereik lezen en schrijven. In feite vindt een soort spiegelproces plaats. Dataformats uit de ene besturingswereld worden in het Dual-Port RAM naar dat van de andere overgeschreven en visa versa.
Na het starten van het programma PCDISK komen de commando's AREAD en AWRITE ter beschikking. AREAD.EXE kopieer dataformats uit het Amiga-deel in PC compatibele formats. AWRITE.EXE kopieert datapartijen van het PC- naar het Amiga-deel. De namen van de te dupliceren files luisteren heel nauwkeurig en jokers (wildcards) zijn bij AREAD niet toegestaan.



De PC-kaarten hebben een veel langere historie dan de Amigakaart

De harddisk

Een AT zonder harddisk is als een auto zonder wielen. Vooruit komen doet het niet erg. Het gebruik van een harde schijf geeft niet alleen meer opslagcapaciteit voor data, maar versnelt ook door verbetering van de lees/schrijfen zoektijden de systeemprestaties aanzienlijk.
Voor de aansluiting van de harddisk zijn er twee mogelijkheden:

- Aansluiting van een *IBM PC compatibel model* op de gecombineerde diskdrive- en harddiskcontroller van de AT-kaart
- Het aanleggen van *MS-DOS partitions* op een Amiga- harddisk. Via het programma AUTOBOOT kun u zo'n DOS-harddisk op een Amiga harde schijf emuleren.

De eerste optie is aantrekkelijk omdat snelle 40 MB (en meer) harde schijven voor de IBM PC AT momenteel spot goedkoop zijn. Bovendien kan de PC harddisk zowel AMIGA-DOS als MS-DOS partitions bevatten zonder dat beide werelden dat van elkaar weten. Andersom is het via JLINK.COM (onder Janus) maximaal vier virtuele MS-DOS drives op één Amiga harddisk aan te leggen. Daarbij is het niet nodig om gescheiden Amiga-DOS en MS- DOS partions op de Amiga harddisk aan te leggen. De emulatiesoftware zorgt er voor de vertaling van het MS-DOS in het Amiga-DOS format.

Welke type harddisk u ook neemt dit randapparaat vergroot de over-all Amiga-prestaties enorm. Lezen/schrijven zoeken gaan vele malen sneller dan bij diskdrives en grote files kunnen zonder problemen worden weggeschreven. Bovendien staan de programmatuur met bijbehorende datafiles en het operating systeem nu op een en het zelfde medium. Dat bespaart veel tijdrovend schijven-gewissel.

De praktijk

Ook in de dagelijkse praktijk rijzen twee interessante vragen. De eerste vraag betreft de *compatibiliteit* van de bestaande MS- DOS software en hardware met de "Amiga AT". Dat viel reuze mee. Alle gangbare toppers zoals Lotus 1-2-3, dBase III, Flight Simulator III, WP 4.X, WordStar en PC Tools liepen als een zonnetje. De MS Mouse wordt via AMOUSE door de Amiga-muis geimmiteerd. Dat werkte bij de geteste tekenprogramma's goed. OS/2 hebben we wegens gebrek aan voldoende on board RAM van onze Amiga 2000 niet kunnen beproeven.
Uiteraard is het ondoenlijk om alle beschikbare PC hardware op de Amiga 2000 met A 2286 AT-kaart te gaan testen. Het werd daarom een at random steek proef. Een als Epson geconfigureerde Star printer en als HP Laserjet Plus geconfigureerde laserprinter gaven over de parallelle poort geen noemenswaardige problemen. Een bekende 28 ms 40 MB harddisk ook niet. Bij een 1200/2400 baud modem onstonden aanvankelijk problemen. Enig gepeuter met de dipswitches en jumpers bracht echter uitkomst.
De twee interessante vraagt betreft de *AT snelheidsprestaties* van het A 2286 bridge-board. Zoals de kloksnelheid van 8 MHz al aangeeft gaat het hierbij niet om een waar snelheidsmonster. Ten opzichte van de 4,77 MHz XT-kaart kan de A 2286 2-5 maal sneller zijn. Ten opzichte van een 10 of 12 MHz AT blijft de Amiga 2000 AT 20-40% achter. Dit alles is uiteraard sterk afhankelijk van de systeemconfiguratie en toegepaste software.

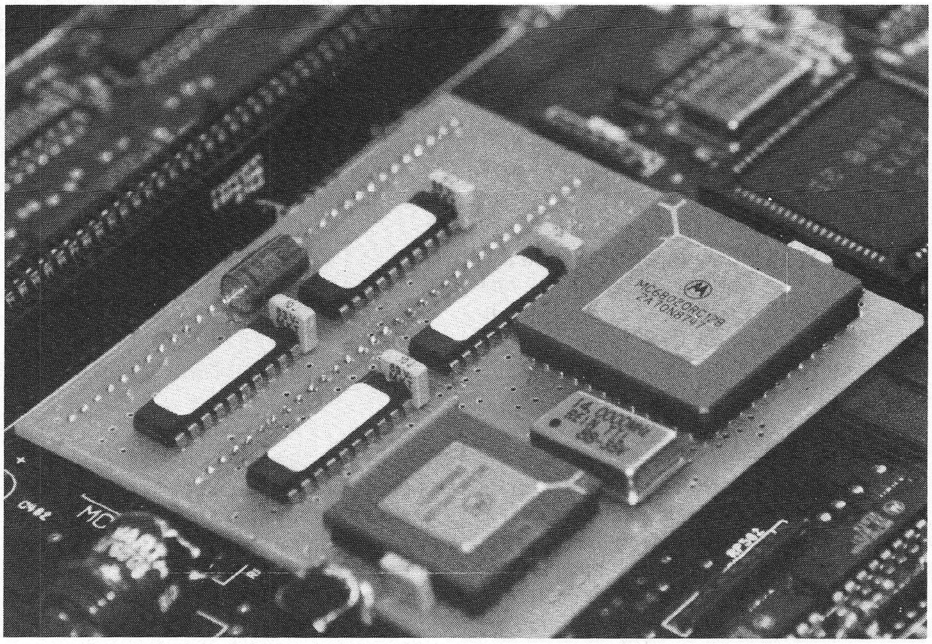
De Amiga als zich zelf en IBM compatibele PC AT biedt "the best from both worlds". Hoewel niet het allerbeste uit de MS-DOS AT- wereld. Wie een supersnelle AT zonder optimale grafische prestaties zoekt is met een PC AT-kloon beter af dan met een Amiga-hybride. Voor de gebruiker die zowel de Amiga-capaciteiten wil benutten als serieus MS-DOS wil gaan draaien biedt de A 2286 bridgeboard uitkomst. Dan is de prijs allezins redelijk. Voor HIRES kleur-, video-, animatie en full stereo-gebruik blijft de Amiga de aangewezen machine.

U.S.

68000 Intern (1)

Cursus programmeren voor de MC68000 processor

Ook zo'n zin in het schrijven van supersnelle routine's waarvan uw vrienden verbaast zullen staan? Al was het alleen maar om op verjaardagen en dergelijke de show te kunnen stelen door te kunnen zeggen: "Ik (m/v)? Ik oriënteer me de laatste tijd voornamelijk op de achtenzestigduizend micro". We zullen u in een aantal korte lessen de kans geven te leren programmeren op een wel heel populaire microprocessor. Een chip die het hart vormt van menig computer. Aan deze cursus kan iedereen deelnemen, alhoewel enige voorkennis van machinetaal geen overbodige luxe is.



De 68000 microprocessor

Menigeen onderons heeft wel eens geprogrammeerd in machinetaal op de Commodore C-64/128. Diegene die zo gefortuneerd waren dat zij zichzelf een Amiga wisten te verschaffen hebben allemaal één ding gemeen: ze kochten de machine vanwege zijn enorme mogelijkheden. Met die enorme mogelijkheden stuitte ze meteen ook op één van de grootste problemen; namelijk dat ze van de machine zelf geen barst meer begrepen. En dan te bedenken hoeveel plezier je kon beleven aan een zelfgeschreven stukje machinetaal-programmatuur. Dat plezier is nog een stuk groter met een machtige microprocessor als de MC68000, daarom raden wij iedereen aan vanaf nu mee te lezen met onze cursus MC68000.

Een familieportret

De Motorola MC68000 is de derde generatie van de Motorola microprocessors. De eerste generatie was de 6800, een 8 bits machine. De tweede generatie was de 6809, die zowel 8 als 16 bits karakteristieken had. De huidige MC68000 is ontstaan in 1979, en draagt de naam MC68000L4. De vier in de type aanduiding geeft de maximale aantal klokcycli per seconden aan, in miljoenen. Evenals de MC68000L6, MC68000L8, MC68000L10 en de MC68000L12 heeft deze 32-bits interne registers, een 16-bits databus, en een 24-bits adresbus. Dan is er sinds 1982 de MC68008, met een 8-bits databus, en intern 32-bits registers. De MC68010 (1982) is een verdere ontwikkeling in de reeks van Motorola. De MC68010 is geschikt voor systemen met een virtueel geheugen. Dat wil zeggen dat u gebruik kunt maken van een reeks programma-adressen die totaal verschillend kunnen zijn van de beschikbare geheugenplaatsen. Zodoende 'ziet' en programmeert de programmeur een virtueel geheugen. De MC68020 heeft een 32-bits databus en een 32-bits adresbus. Hiermee is de MC68020 een complete 32-bits CPU geworden. Opgemerkt moet worden dat de standaard MC68000 intern ook echte 32 bits registers heeft. De MC68020 is geschikt voor het samenwerken met de MC68881 floating-point coprocessor. Deze neemt het werk van de microprocessor over bij taken die normaal door rekenkundige software worden uitgevoerd (en dat kost CPU-tijd). Daarbij heeft de MC68020 een 'instruction cache', een stukje geheugen in de CPU-chip waarin van te voren, nog voor ze aan de beurt zijn, een aantal instructies worden gelezen. Dit verhoogt de verwerkings snelheid van de CPU.

De MC68000 is ontworpen met de behoefte aan een meer op een hogere programmeertaal afgestemde CPU. Althans meer dan die van een assembly programmeur. Hij maakt ook beter gebruik van statistische informatie over programma's dan vroegere microprocessors. Een leuk voorbeeld is het feit dat kleine constanten vaker voorkomen dan constanten met een hoge waarde. Daarom heeft de MC68000 een aantal speciale korte instructie's voor het werken met 8-bit constanten. Hiernaast zijn er ook nog speciale instructie's die het werken met procedure's eenvoudig maken. Een nadeel van de MC68000 is dat het geen floating-point instructie's kent, dus er zijn geen floating-point registers. Als we dan toch floating-point berekeningen willen maken, zullen we een aparte floating-point processor aan onze computer moe-

ten toevoegen. U zult later ontdekken dat er ook software-matig oplossingen voor te vinden zijn.

De behuizing in de AMIGA

Omdat de meeste onder ons nog steeds op het uiterlijk af gaan, hier eerst een blik op de buitenkant van de MC68000.
De MC68000 is verkrijgbaar in een aantal behuizingen, met verschillende vormen en materialen. De MC68000 heeft 64 aansluitingen voor de verbinding van de chip met de externe bouwstenen. De MC68020 (32 bits) heeft meer aansluitingen, en de MC68008 (8 bits databus) heeft er minder. De standaard behuizing van de MC68000 is de DIP. Dat wil dus zeggen 64 pinnen die in twee rijen naast elkaar staan. Laten we eens een vluchtige blik werpen op de Commodore Amiga. In de Amiga vinden we, zoals bekend verondersteld wordt, ook de MC68000. Er is een adresbus, en een databus op aanwezig. In de Amiga zijn de besturingslijnen van de asynchrone bus verbonden met een IC dat de naam 'Gary' draagt. 'Gary' is een bus control chip, die op zijn beurt weer sterk samenwerkt met 'Fat Agnus'. Deze chips zorgen voor de synchronisatie van signalen, en controleren tevens of er geschreven mag worden naar de diverse geheugengebieden in de Amiga. De dynamische ram IC's in de Amiga moeten worden gerefreshed, dit wordt ook door 'Fat Agnus' gedaan. 'Fat Agnus' is niet zo Fat als de naam doet vermoeden, slechts 20 adreslijnen zijn intern aanwezig. Dit heeft erin geresulteerd dat de Amiga maximaal 1M byte 'onboard' heeft. De MC68000 kan door het uitblijven van het DTACK signaal even vast worden gehouden door 'Gary'. Hier moeten we het voorlopig even bij laten, want dieper op deze materie ingaan is voor deze cursus absoluut overbodig. Voor mensen die iets meer hiervan willen weten verwijs ik allereerst naar het handboekje dat bij uw computer werd geleverd. In de Appendix bevindt zich een beknopte technische beschrijving van de Amiga.

MC68000 registers

In de MC68000 bevinden zich een aantal geheugenplaatsen die we registers noemen. In deze registers bergen we tijdelijk informatie op die nodig is voor het uitvoeren van de instructies. Een aantal van deze registers zijn bestemd voor intern gebruik, en een aantal zijn zogenaamde programmeerbare registers. Deze laatste zijn toegankelijk voor de programmeur met behulp van instructies. Zo kennen we 8 dataregisters (32-bits) en 9 adresregisters (32-bits). Dit zijn de 'general purpose' registers. Er is een *program counter* (32-bits), deze wijst op de eerst volgende instructie in het geheugen die opgehaalt moet gaan worden. Het is goed als er veel interne registers in een CPU zijn. Bewerkingen op registers onderling hebben het voordeel gegevens niet uit een extern geheugen te hoeven halen. Hierdoor neemt de verwerkingssnelheid toe. Twee van de adresregisters dragen de naam *User Stack Pointer [USP] en Supervisor Stack Pointer [SSP]*. Dit zijn wijzers naar het stapelgeheugen voor programma's in user mode, en programma's in supervisor mode (zoals systeemsoftware). In usermode is alleen de USP aktief en beschikbaar voor de CPU. In supervisormode gebruikt de CPU de SSP voor het opslaan van de returnadressen bij een interrupt of een subroutine. De CPU kan desgewenst ook de USP lezen of veranderen. Wat is nou het voordeel van twee stackpointers? Doordat programma's in usermode draaien blijft de supervisor stack onaangeroerd. Systeemsoftware is dus beschermd tegen inbreken op de supervisor stack. A7 is het adresregister dat dienst doet als *system stack pointer*. Dan is er de *program counter*, die kan 2^23 woordplaatsen (16-bits) in het geheugen adresseren. Dat is meer dan 8 miljoen words. Omdat extern niet alle adreslijnen op de MC68000 aanwezig zijn, is het volledige 32-bits adrescapaciteit niet beschikbaar. Dan is er als laatste nog het statusregister. Deze geeft de toestand van de CPU aan, en de conditiecode's.

De adresregisters

De MC68000 heeft negen adresregisters. Het adresregister is maximaal 32-bits breed. Het adres waar deze op wijst heeft dus maximaal de waarde 2^32. Dat zou in principe betekenen dat door u ruim 4 miljard geheugenplaatsen kunnen worden geadresseerd. De makers van de MC68000 hebben echter gedacht dat u met de 24 laagste bits wel genoegen zou nemen. Daarom zijn slechts 24 bits bij adressering van externe geheugenplaatsen aktief. Hierdoor is de effectieve adresseringscapaciteit dus (nog maar) 16 MByte. In de negen adresregisters passen uitsluitend words (16-bit) of longwords (32-bit). Zeven van deze registers [A0..A6] worden zowel gebruikt door programma's in user mode als door programma's in supervisor mode. Alleen één van de twee overige registers kan als A7 gebruikt worden door programma's. Dit zijn de twee registers die dienen als *stack pointer*. Hiervan is één voor de user mode, en de ander voor de supervisor mode. Zo'n stack is een aaneengesloten geheugengebied. De geheugenplaatsen worden aangewezen door een *stackpointer* (engels voor stapelwijzer). De eerste zeven registers [A0..A6] kunnen eventueel door u gebruikt worden als waren het stack pointers. Zo kunt u dus eigenhandig een aantal stacks creëren.
De stack pointer (SP) verteld ons waar de top van de stack is. De bodem ('bottom') van de stack is een vast adres, maar hier hoeven we geen rekening mee te houden. Stacks kunnen zowel van hoge adressen naar lage lopen, alswel andersom. In de CPU zijn een aantal specifieke stack operaties aanwezig. Twee van de belangrijkste zijn PUSH X en POP Y. Bij PUSH zal de stackpointer veranderen, en wijzen op een volgend adres. De waarde X zal nu opgeborgen worden in het geheugen op de plaats aangewezen door de stack pointer. Met een PUSH zal de stack met één item groeien. POP Y (PULL Y) doet exact het omgekeerde. POP Y doet de stack met één item afnemen, door het top item op de stack in Y te stoppen. Daarna wordt het item van de stack verwijderd door de stack pointer met de grootte van het item te veranderen. Een andere operatie die met de stack kan worden uitgevoerd, is het veranderen van de stack pointer zonder dat er data op de stack wordt geplaatst. Dit wordt normaal gesproken gedaan als we in een hogere programmeertaal (zoals pascal) een functie of procedure in gaan. Hiermee scheppen we ruimte voor lokale variabelen.
Een programma kan elk adresregister van de MC68000 gebruiken als stack pointer door de bij stack-operaties behorende adresseringsmethodes te gebruiken. Daarbij wordt de waarde in het adresregister, dat als stack pointer gebruikt wordt, bij een PUSH- of POP-bewerking automatisch met het juiste aantal verhoogd of verlaagd. Stacks die gebruik maken van A0..A6 als stack pointer, kunnen uit bytes, words (2-byte) of longwords (4-byte) bestaan. Tot zover even het stack verhaal, adresregisters kunnen ook worden gebruikt als *indexregisters*. De indexwaarde wordt opgeteld bij de inhoud van een ander adresregister, om op deze manier het effectieve adres van een operand in het geheugen te berekenen.

Dataregisters

De MC68000 heeft acht dataregisters [Dn=0..7]. De microprocessor zorgt ervoor dat dat operands kunnen worden opgevat als byte (Dn)[7:0], als word (Dn)[15:0] of als longword (Dn)[31:0]. In de instructie die een dataregister gebruikt, moet worden aangegeven wat de lengte is die bedoelt wordt. Dat deel van het betreffende dataregister kan gebruikt worden als *accumulator*, als *buffer*, als *tijdelijk register* of als *indexregister*. Als je berekeningen wilt uitvoeren met de inhoud van de dataregisters, dan moet je de dataregisters opvatten als accumulators. Deze bevatten dan de operands van aangegeven lengte. De dataregisters zijn een

buffer voor de data tussen databus en de processor. Deze buffer is nodig omdat de MC680000 een 16-bits databus heeft. Hierdoor is het mogelijk om in één keer 8 of 16 bits over te brengen, maar niet een 32-bits waarde. Een willekeurige 32-bits waarde moet in twee maal 16-bit over de databus worden gebracht. De dataregisters worden ook nog toegepast als tijdelijke opslag van data uit andere registers, of van de processor. Net als bij de adresregisters het geval was, kunnen dataregisters ook als *indexregister* dienen. De inhoud van een dataregister kan worden opgeteld bij de waarde van een adresregister, op deze manier berekenen we het effectieve adres van de operand. Het aardige van de indexadressering is dat de index eenvoudig gewijzigd kan worden met behulp van elke instructie die op een dataregister kan worden toegepast. Hiermee is het mogelijk geworden dat de index tijdens het doorlopen van een programma op een slimme manier veranderd. Deze methode zien we o.a. toegepast bij 'loops'.

Zo nog steeds niet in slaap gevallen? Het lijkt voor velen eerst nog een onsamenhangende rotsooi. Maar voor degene die al eerder iets met machinetaal hebben gedaan zal het best wel meevallen.

```
Afbeelding 1.
Het formaat van een dataregister in de MC68000:
---------------------------------------------------------
-----------------LongWord-------------------
                                       --------Word--------
                                                  ----Byte---
31                                    16 15         8 7      0
=============================================================
```

Tot slot

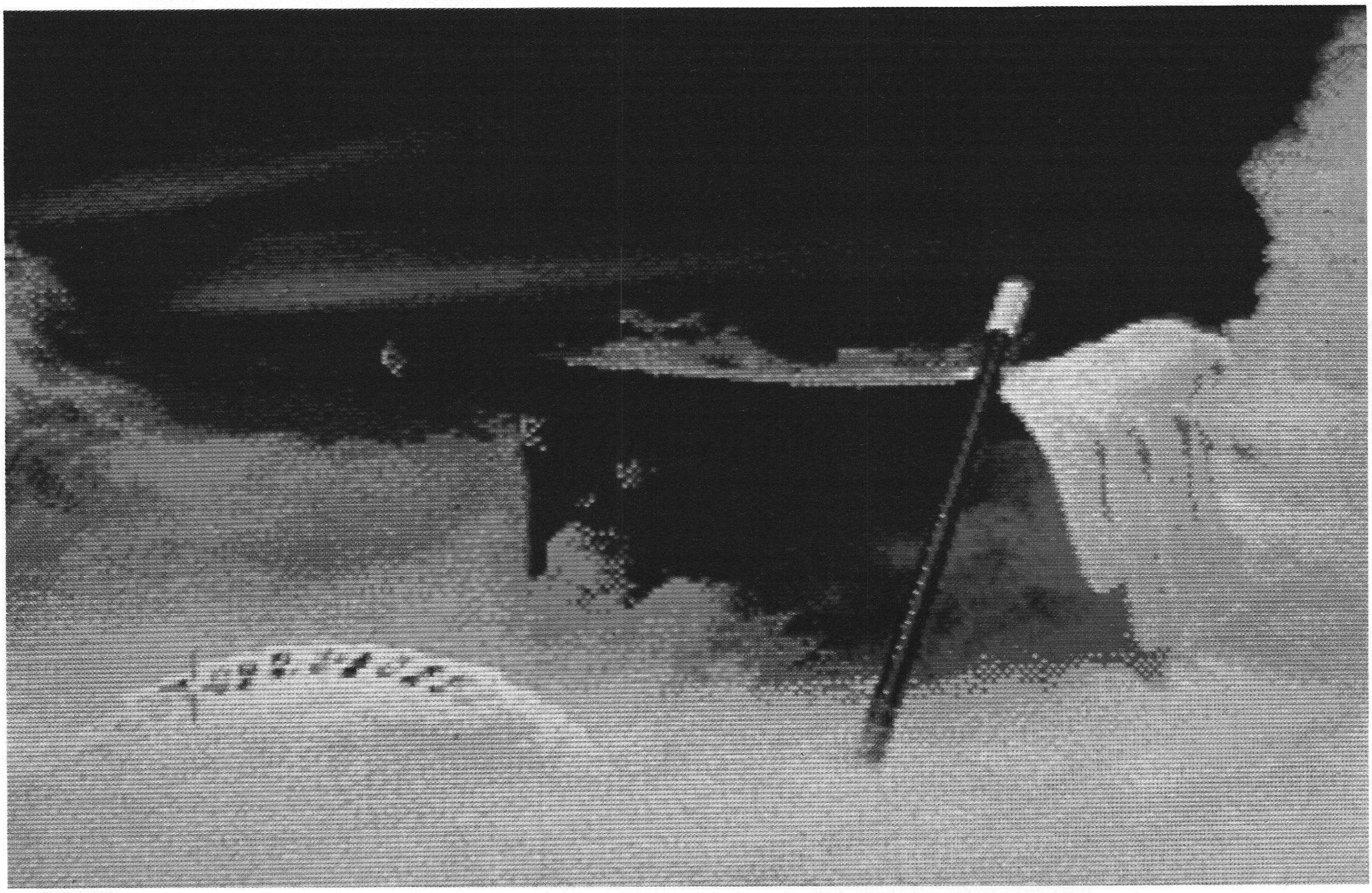
We zijn al weer aan het eind gekomen van deze inleidende stof tot het werken met de MC68000 voor deze keer. We hebben u nog niet heus kennis laten maken met de machinetaal, maar dat komt nog. Weest u ervan bewust dat juist voorgaande zaken van primair belang zijn bij het werken met onze computer. Soms is het handig als u weer even terugblikt op voorgaande zaken (doe dat ook!).

De volgende keer laten we u uitgebreid met de stack pointer kennis maken, tevens komen nog de program counter als wel het status register aanbod. De uiterst krachtige instructieset van de MC68000 zal dan ook stapsgewijs besproken worden.

Johan & johan

Animagic

"Toveren" met ANIMS en .IFF-files



De naam Animagic suggereert het "toveren" met animaties. Hoewel dat toveren misschien iets overdreven is kan dit Aegis-pakket toch verbluffende animatie-effecten uit de Amiga trekken. Een prima pakket dat uw desktopvideo-, bedrijfs-, produkt-, educatieve en muziekpresentaties een professioneel tintje meegeeft.

Een betrekkelijk nieuwe ontwikkeling bij de desktopvideo op de Amiga is de Special Effect Generator (SEG)-software. Onder SEF verstaat men allerlei grafisch trukages zoals kleurveranderingen, rotatiebewegingen, schaduweffecten, stroboscoop, in/uitzoomen, wipes, fade in/out. Deze speciale effecten worden gebruikt om videoproducties, TV-shows, kabellogo's en zakelijke presentaties meer impact te geven. Ook wordt grafische SEG-software dikwijls bij de productie van tekenfilms ingezet. Met Aegis Animagic kunt u dat ook allemaal zelf op de Amiga.

De hoge hoed met kaartspel, witte handschoenen toverstokje op de verpakking van Aegis Animagic wekt de illusie met een goochelaar van doen te hebben. In dit SEG-pakket schuilt de magie hem echter meer in de digitale video-effecten, flitsende animaties en editingmogelijkheden dan het beduvelen van het publiek. Animagic kan veel met .IFF-bestanden en ANIM-style animaties. Fonteinen, flips, schaduwen, kleureffecten en -cycling, stroboscoop, draadmodellen, diverse editing faciliteiten, noem maar op en het staat op de diskettes. De vakman en gevorderde video-amateur kan er direkt meé aan de slag.

Dat gevorderde staat hier niet voor niets. Ondanks alle hulp die de Amiga en software de gebruiker bieden moet je wel enig inzicht in de bruikbaarheid van de geboden effecten hebben. Anders ontstaat en een niet te volgen of irriterende effectwarboel. In de beperking toont zich de meester. Alleen daar effecten aanbrengen

waar ze ook echt nodig zijn. Anders ontaarden de effecten in een maniertje om de eigen onkunde te verdoezelen.

Wat doet Animagic?

Aegis *Animagic* is een kruising tussen een dtv video editing console en een SEG. In eenvoudig nederlands betekent dat een kombinatie van een videosnijtafel en een generator voor speciale effecten.
De *snijtafel* helpt u bij het samenstellen van de animaties. Dat kan als korte losse beeldseries of als komplete tekenfilms. Die tekenfilms bestaan dan uit aan elkaar gelaste losse animatie- sequenties. Tot de beschikbare editopties behoren o.a. het toevoegen of verwijderen van beelden (= frames), cut en paste, het aanpassen van het kleurenpalet en de timing van de beeldsequenties.
De software *Special Effect Generator* (SEG) brengt via de raster mapping-techniek speciale effecten aan. Vroeger moest zoiets met peperdure hardware gebeuren. Nu "tovert" u voor *f* 269,- de fraaiste Digitale Video Effects *(DVE's)* uit de Amiga. Volgens de handleiding zijn er op elke DVE zo'n 9025 variaties mogelijk. Ons ontbrak het uiteraard aan de tijd om al deze kombinaties te gaan beproeven. Wel raakten we bij de geteste mogelijkheden onder de indruk van de SEG-kracht en het relatieve bedieningsgemak van Animagic.

De Animagic-studio

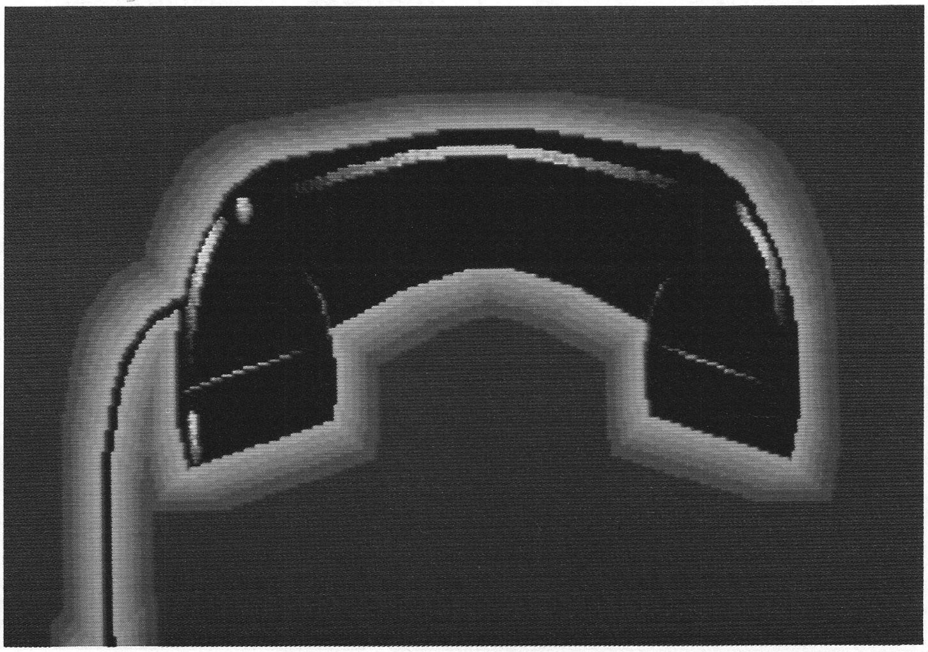
Door het starten van de *Edit Bay Main Control panel* verandert de Amiga in een echte DTV-studio. Via dit controlepaneel kunt het gehele video-editproces besturen. Alle belangrijke functies zoals het inladen van de frames, cut en paste, het aangeven van de tijdschaal (met start en stop van de animaties), het instellen van het kleurenpalet en de color-cycling, het opgeven van de videobron- (maximaal 2) en videobestemmingsbuffers. Verder behandelt de Edit Bay Main Control panel de screen size, het maken van eindloze animaties (looping), het over elkaar heen leggen (layer) van ANIMS en het instellen van de schermresolutie met bijbehorende kleurmodus.
Alle functies zijn volledig muisgestuurd en de help-sequesters kunnen overal worden opgeroepen.

ANIMS

Alvorens naar de DVE-opties te gaan eerst iets over de ANIM- techniek. ANIM is een standaardfileformaat voor animaties en werd ontwikkeld door Aegis en Sparta Inc. De bedoeling van het ANIM-formaat is om de omvangrijke animatiesequenties zo kompakt mogelijk op te slaan en bij het oproepen weer snel af te spelen.
Een ANIM zit alsvolgt in elkaar: Als eerste wordt een standaard .IFF grafisch bestand van beeld 1 gemaakt. Van het daarop volgende beeldje 2 behoeft de Amiga slechts te weten in hoeverre dit beeldje van het eerder opgeslagen frame 1 afwijkt. Alleen de informatie over de verschillen van de opeenvolgende beeldjes wordt naar de schijf weggeschreven. Dat bespaart enorm veel ruimte in vergelijking met het in zijn geheel saven van deze iets gewijzigde frames. Een animatiesequentie van vele MegaBytes kan zo met gemak tot minder dan één MB worden teruggebracht.
Natuurlijk hangt de uiteindelijke grootte van het ANIM-bestand af van het aantal veranderingen dat tussen de opeenvolgende schermen optreedt en de gebruikte resolutie. Hoe meer veranderingen en hoe meer pixels des te groter de ANIMs.
Het ANIM-formaat wordt momenteel toegepast in Aegis VideoScape 3D,, Aegis Modeler 3D, Aegis Video Titler, Lights!, Camera!, Action!, Deluxe Paint III, The Director, Phonton Cel Animator, Hash Animation Series en utilities zoals GrabANIM en ShowANIM.

DVE Control Window

Het kontrolevenster voor de DVEs vormt de sleutel tot hoge hoed vol met truuks van Animagic. De gebruiker zoekt een geschikt DVE uit het popup-menu. N.B.: De Digitale Video Effecten staan in DVEMaps directory op de programmadiskette en zijn herkenbaar aan de toevoeging .MAP. Is er eenmaal voor een bepaald DVE gekozen dan gaat u samen met Animagic bekijken hoeveel frames er voor het beoogde DVE nodig zijn. Via PREVIEW en MANUAL PREVIEW kunt u de werking van het DVE bij het opgegeven aantal beeldjes bestuderen. Dat kan in voor- en achterwaartse richting.
Zoals reeds eerder vermeld werd kan de videomaker uit twee videobronnen (Source1 en Source2) kiezen. Alleen de opgegeven *S* wordt in bewerkte vorm naar de bestemmingsbuffer gekopieerd. De andere S blijft voorlopig ongewijzigd.



Verder bevat het DVE kontrolevenster nog instellingen voor TRANSLUCENT (verloop stroboscoop van gekleurd tot geheel doorzichtig), Outline, Backdrop, Grid en 3D.Welke effecten kunt u nu met de DVE.MAPs maken? Even afgezien van de 9025 variaties zijn devolgende SEGs mogelijk:

- **Expand**. Het groter worden van een object vanuit een opgegeven schermpunt
- **Horizontale Blind**. Rotatie om de horizontale as. Gebruikt voor 3D-effecten
- **Spin1**. Laat het beeld van uit het centrum tot schermvullend uitdeien met daarin opgenomen één omwenteling
- **Spin3**. Idem met drie omwentelingen
- **Split** verdeelt het beeld in twee delen. De rechter beeldhelft begint links en de linker beeldhelft start rechts.
- **Squeeze** drukt het huidige beeld naar rechts samen terwijl het volgende beeld vanuit links wordt uitgerekt


Charlie Chip's
DOOR:
SOFT WIRWAR
JAWEL.. DIE ADVERTENTIE HEB IK GEZET!
DUS JE KOMT?!
VAN-AVOND NOG!?
DE KRANT
MOOI... NOU... TOT STRAKS DAN!!
!
WAT WAS DAT?
EEN HACKER!
EVEN LATER..
"HIER MOET HET EEN EN ANDER GEKRAAKT WORDEN"?!
KOM BINNEN!
NOU.. HIER STAAT ALLES DUS!!
MOOI!! DAT ZIET ER GOED UIT!
BEGIN MAAR VAST.. DAN ZET IK EVEN KOFFIE!!
HAK!
HAK!
HAK!
HAK!
HM... HIJ LAAT ER INIEDERGEVAL GEEN GRAS OVER GROEIEN!..
ZO!... DE KOF.....FIE...
HAK!!
KRAK!
WAT MAAK JIJ NOU IDIOOT!!
NOEM JE DAT HACKEN?!!
!
HACKEN?!!
WAT IS DAT?!.. IK BEN HAKKER!
EIGENLIJK ALLEEN MET HOUT.. MAAR MET COMPUTERS KAN 'T NATUURLIJK OOK!..

- **Vertical Blind** voor rotatie om de vertikale as en het creëeren van 3D-effecten
- **Wrap.** Iris/diafragma-effect, deelt het beeld in vier stukken op en sluit net als een diafragmasluiter bij een fotokamera
- **Box2** plaatst het beeld uit S1 op een zeshoekige doos, draait de doos over 90 graden en laat S2 op het nu voorliggende vlak zien
- **Box4** laat S1 op de vier zijden van een draaiende (vertkale of Y-as) doos zien
- **Confetti** het beeld regent in 225 deeltjes van links boven als een douche over het scherm
- **Corner Expand** trekt het beeld vanuit de linker bovenhoek over het gehele scherm uit
- **Doors10** opent tien verticale jalouzieën en toont de daarachter gelegen S2
- **Doors2** opent als een tweedelige deur om de daarachter gelegen S2 te laten zien
- **Dribble** trekt een druppelend verfspoor van 160 vertikale paintstrepen
- **FallAway5x5** laat 25 beeldfragmenten naar hun beeldpositie draaien om één geheel beeld op het scherm te zetten
- **Horizontale Blind6** geeft zes horizontale jalouzieën die van boven naar beneden steeds sneller draaien
- **Page_Turn** draait het getoonde beeld als een boekpagina om
- **Plain** is de standaard DVE.MAP in kopieert het volledige ongewijzigde source beeld uit S1 of S2 naar de doelbuffer
- **SlideIn** doet een klein beeldje dat steeds groter wordt en recht op gaat staan het scherm glijden
- De **Tumble** is een bekende SEQ die S1 over de horizontale as doet tuimelen en vervolgens door S2 vervangt.
- **Unfold9** vouwt negen beelddelen als een waaier uit tot één beeldvullend geheel
- **Vertikale Blind6** gebruikt zes vertikale jalouzieën met een oplopende symmetrische rotatie

Al deze standaard effecten kunnen via het kontrolepaneel bewerkt worden. Bijvoorbeeld met stroboscopisch, verandering van kleurpalet of schaduwwerking.

Hardware en handboek

DTV en SEGs stellen hoge eisen aan de Amiga. Voor minimaal gebruik is een Amiga met 1MB aan vrij RAM voldoende. Wie het onderste uit de kan wil doet er beter aan om 2MB aan RAM en een mathematische coprocessor of een MC 68020 te installeren. Animagic is van huisuit al berekend om met een MC 68020 of MC 68881 te werken.
Het meegeleverde engelse Aegis-handboek is ons inziens wat beknopt. De beginner kan na het lezen van de tekst en het oefenen met de tutorials wel aan de slag, maar creatieve suggesties ontbreken. Dat is toch jammer gezien de grote potentiële mogelijkheden van Animagic. De gevorderde videohobbyist en professional zullen hun weg wel vinden.

Animagic is een goedkope en veelzijdige oplossing voor een aanzienlijk duurdere hardware SEG-generator. Echt professionele SEGs bieden meer. Zij kosten echter het twintigvoudige en zijn minder gebruiksvriendelijk dan dit Aegis DTV-pakket.
Info: Altycos, tel.: 079-51075.

U.S.